**DISSERTATION**
**for the degree of**
**Doctor of Natural Sciences (Dr. rer. nat.)**

# The Role of Active Learning in Developing Trustworthy AI

## Approaches for Enhancing Transparency and Explainability in Processes and Systems

Fabian Stieler

2024

University of Augsburg
Department of Computer Science
Software Methodologies for Distributed Systems

**The Role of Active Learning in Developing Trustworthy AI**
Approaches for Enhancing Transparency and Explainability in Processes and Systems

**Supervisor**
Prof. Dr. Bernhard L. Bauer
Department of Computer Science
University of Augsburg, Germany

**Advisors**
Prof. Dr. Jörg Hähner
Department of Computer Science
University of Augsburg, Germany

Prof. Dr. Matthias Tichy
Institute of Software Engineering and Programming Languages
Ulm University, Germany

**Day of Defense**
December 19, 2024

**Transparency Statement**
The readability of this text has been improved with the support of AI technologies.

**Copyright**

# Abstract

In today's world, the significance of Artificial Intelligence (AI) is continuously growing as it increasingly permeates various aspects of everyday life, exerting a significant influence on sectors such as healthcare, finance, and other safety-critical domains. As AI systems become more deeply integrated into society, there is an increasing demand for systems that are functionally intelligent and ethically sound. This makes the development of trustworthy AI a societal imperative that aims to align technological advancements with ethical standards and public expectations. In 2019, the European Commission released a framework for trustworthy AI, defining ethical principles and key requirements (High-Level Expert Group on AI 2019).

At the core of AI systems are algorithms trained through machine learning (ML). These algorithms analyze data and derive decisions that can be deployed to solve complex problems. Simultaneously, to realize these algorithms, highly complex models, such as deep neural networks, are often used, whose inner workings can no longer be intuitively understood by humans. This contrasts with the demand for explainability and transparency in trustworthy AI. Moreover, training these models may require large amounts of labeled data, necessitating a laborious and costly data annotation process, especially in high-risk domains. In this context, the method of active learning (AL) proves to be promising, as human experts are involved in the training process to label unknown data.

However, developing AI systems using AL presents challenges due to the dynamic evolution of the data landscape and the high resource demands associated with creating and managing training data. Different methodological approaches from software engineers and data scientists necessitate revising traditional process models to meet the specific requirements of high data variability due to continuous model retraining. Effective harmonization of these diverse development approaches is essential to ensure efficient and collaborative workflows.

To address this, this dissertation presents a revised life cycle model that combines agile principles with the requirements of data-driven AL projects. It aims to enhance the efficiency and transparency of development by structuring the entire development cycle from planning to implementation and facilitating collaboration within heterogeneous teams. Central to this is an innovative development methodology that encompasses principles for data, code, and automation and provides a workflow that guides teams in establishing a robust development process during the implementation and operation of the AI system. This methodology promotes agility and collaboration while ensuring adherence to best practices.

Another crucial element for developing trustworthy AI is traceability, which, from a technical perspective, extends from the original data sources to the model outputs. This includes both user inputs in the form of annotations and model-generated predictions. Traditional frameworks often prove inadequate in this regard, as they primarily focus on executing the AL methodology and neglect aspects of traceability. There is a clear need for a framework that integrates both AL functionality and the necessary components for capturing data provenance and versioning of artifacts throughout the life cycle, thereby contributing to the reproducibility of results.

To address this challenge, this dissertation presents a new open-source framework called LIFEDATA, designed to facilitate end-to-end traceability and efficient data annotation in AL projects. LIFEDATA provides a structure for integrating technical components essential for traceability, including versioning of code, data, and models, as well as logging user interactions. The broad applicability of the framework is demonstrated through two use cases in the life sciences, including the analysis of skin images and the classification of ECG signals. These use cases illustrate how AL can improve both the efficiency of data annotation and the quality of model results.

Moreover, the explainability of AI systems is crucial for achieving transparency in technical processes and systems. However, as models become increasingly complex, the challenge of making them understandable to humans grows, with concepts of eXplainable AI (XAI) playing a central role. An integrative approach

is required, one that applies interpretable ML (IML) methods tailored to the specific needs of various stakeholders.

To this end, a domain-specific approach specifically tailored to the use case of skin image analysis will be presented. By combining a human-centered approach to the classification of skin lesions with an interpretation method for AI-based systems, human-understandable explanations of model outputs are examined. Additionally, the XAI-Compass is introduced, a concept for involving diverse stakeholders in the development process. This instrument helps identify critical contact points between humans and model interpretations through the systematic organization of roles, life cycle phases, and goals, allowing explanations of AI systems to be tailored to the specific needs of different stakeholders. Three studies were conducted using the ECG signal classification use case to develop and evaluate explanations for various stakeholder groups.

Finally, this dissertation demonstrates that the proposed development life cycle and methodology for AL projects have the potential to significantly enhance both transparency and collaboration in the development of trustworthy AI systems. The introduced LIFEDATA framework, with its modular structure and focus on artifact traceability, proves to be a practical solution for implementing AL across various application areas, as evidenced by its successful use in the life sciences. Furthermore, the domain-specific implementation of an IML method provides an innovative approach that helps make the outputs of an AI system understandable to humans. In connection with integrating stakeholder perspectives through the XAI Compass, the conducted studies show that involving different stakeholders improves the acceptance of model interpretations and helps make them more understandable and relevant to various user groups.

Overall, this dissertation presents new approaches and practical solutions to the challenges in developing trustworthy AI systems with AL, by focusing on transparency, traceability, and explainability.

# Zusammenfassung

In der heutigen Welt nimmt die Bedeutung der Künstlichen Intelligenz (KI) kontinuierlich zu, da sie zunehmend in zahlreiche Lebensbereiche vordringt und signifikanten Einfluss auf Sektoren wie das Gesundheitswesen, die Finanzbranche und weitere sicherheitsrelevante Domänen ausübt. Mit der wachsenden Integration von KI-Systemen in den gesellschaftlichen Rahmen wird der Ruf nach Systemen, die nicht nur funktional intelligent, sondern auch ethisch vertretbar sind, immer dringlicher. Dies macht die Entwicklung vertrauenswürdiger KI zu einem gesellschaftlichen Imperativ, der technologische Fortschritte mit ethischen Standards und öffentlichen Erwartungen in Einklang bringen soll. Die Europäische Kommission hat daher im Jahr 2019 ein Rahmenwerk für vertrauenswürdige KI veröffentlicht, das ethische Prinzipien und Schlüsselanforderungen definiert (High-Level Expert Group on AI 2019).

Im Zentrum vieler KI-Systeme stehen Algorithmen, die durch maschinelles Lernen (ML) trainiert werden. Diese Algorithmen analysieren Daten und leiten daraus Entscheidungen ab, die zur Lösung komplexer Probleme eingesetzt werden können. Gleichzeitig werden zur Realisierung der Algorithmen häufig hochkomplexe Modelle, wie tiefe neuronale Netze, eingesetzt, deren innere Funktionsweise von Menschen nicht mehr intuitiv verstanden werden kann, was im Widerspruch zur Forderung nach Erklärbarkeit und Transparenz in vertrauenswürdiger KI steht. Zudem werden für das Training dieser Modelle große Mengen gelabelter Daten benötigt, was insbesondere in risikobehafteten Domänen einen aufwändigen und kostenintensiven Datenkennzeichnungsprozess erfordert. Hierbei erweist sich die Methode des aktiven Lernens (AL) als vielversprechend, da sie die Einbindung menschlicher Experten zur Kennzeichnung unbekannter Datenpunkte ermöglicht.

Die Entwicklung von KI-Systemen mit AL-Methodologie bringt indes Herausforderungen mit sich, da diese Projekte durch die dynamische Evolution der Datenlandschaft und die hohen Ressourcenanforderungen bei der Erstellung und Verwaltung von Trainingsdaten geprägt sind. Unterschiedliche methodische Ansätze von Softwareentwicklern und Data Scientists erfordern eine Überarbeitung traditioneller Prozessmodelle, um den spezifischen Anforderungen der hohen Datenfluktuation durch das kontinuierliche Neutraining der Modelle gerecht zu werden. Eine effektive Harmonisierung dieser verschiedenen Entwicklungsansätze ist entscheidend, um eine effiziente und kooperative Arbeitsweise zu gewährleisten.

Zu diesem Zweck wird in dieser Dissertation ein überarbeitetes Lebenszyklus-Modell vorgestellt, das agile Prinzipien mit den Anforderungen datengetriebener AL-Projekte verbindet. Es zielt darauf ab, die Effizienz und Transparenz der Entwicklung zu erhöhen, indem es den gesamten Entwicklungszyklus von der Planung bis zur Umsetzung strukturiert und die Zusammenarbeit in heterogenen Teams erleichtert. Im Mittelpunkt steht eine innovative Entwicklungsmethodik, die Prinzipien für Daten, Code und Automatisierung umfasst sowie einen Workflow bietet, der als Leitlinie die Teams dabei unterstützt, einen robusten Entwicklungsprozess während der Implementierung und dem Betrieb des KI-Systems zu etablieren. Dabei zeigt sich zum einen, dass die Methodik die Zusammenarbeit und Agilität fördert und zum anderen zur Einhaltung von Best Practices beiträgt.

Ein weiteres wesentliches Element für die Entwicklung vertrauenswürdiger KI ist die Nachvollziehbarkeit, die, technisch gesprochen, von den ursprünglichen Datenquellen bis hin zu den Modellausgaben reicht. Diese umfasst sowohl Benutzereingaben in Form von Annotationen als auch die von Modellen generierten Vorhersagen. Traditionelle Frameworks erweisen sich in diesem Zusammenhang häufig als unzureichend, da sie sich primär auf die Ausführung der AL-Methodologie konzentrieren und die Aspekte der Nachvollziehbarkeit vernachlässigen. Hier besteht ein klarer Bedarf an einem Framework, das sowohl die AL-Funktionalität als auch notwendige Komponenten zur Erfassung der Datenherkunft und Versionierung von Artefakten entlang des Lebenszyklus integriert und damit die Reproduzierbarkeit von Ergebnissen ermöglicht.

Um diese Herausforderung zu adressieren, wird in dieser Dissertation ein neuartiges Open-Source-Framework vorgestellt, das die End-to-End-Nachvollziehbarkeit und effektive Datenannotation in AL-Projekten unterstützt. Dieses Framework bietet eine Struktur für die Integration technischer Komponenten, die für die

Nachvollziehbarkeit erforderlich sind, einschließlich der Versionierung von Code, Daten und Modellen sowie der Protokollierung von Benutzerinteraktionen. Die breite Anwendbarkeit des Frameworks wird durch zwei Anwendungsfälle aus den Lebenswissenschaften demonstriert, darunter die Analyse von Hautbildern und die Klassifikation von EKG-Signalen. Diese Anwendungsfälle verdeutlichen, wie AL dazu beitragen kann, sowohl die Effizienz der Datenannotation als auch die Qualität der Modellergebnisse zu verbessern.

Darüber hinaus ist die Erklärbarkeit von KI-Systemen entscheidend, um Transparenz in technischen Prozessen und Systemen zu erreichen. Mit der zunehmenden Komplexität von Modellen steigt jedoch die Herausforderung, diese Modelle für Menschen nachvollziehbar zu gestalten, wobei in diesem Zusammenhang Konzepte der Erklärbaren KI eine zentrale Rolle spielen. Aufgrund der vielseitigen Anwendungsgebiete und der unterschiedlichen beteiligten Personen ist hierbei ein integrativer Ansatz erforderlich, der Methoden des interpretierbaren ML anwendet und diese auf die spezifischen Bedürfnisse verschiedener Interessengruppen zuschneidet.

Im Rahmen dieser Dissertation wird hierzu ein domänenspezifischer Ansatz entwickelt, der auf den Anwendungsfall der Analyse von Hautbildern zugeschnitten ist. Durch die Kombination einer menschlichen Herangehensweise zur Klassifikation von Hautläsionen mit einer Interpretationsmethode für KI-basierte Systeme werden menschenverständliche Erklärungen der Modellausgaben untersucht. Darüber hinaus wird der XAI-Compass eingeführt, ein Konzept zur Einbindung diverser Interessengruppen in den Entwicklungsprozess. Dieses Instrumentarium hilft, durch die systematische Organisation von Rollen, Lebenszyklusphasen und Zielen wichtige Schnittstellen zwischen Menschen und Modellinterpretationen zu identifizieren, um die Erklärungen von KI-Systemen gezielt auf die spezifischen Bedürfnisse unterschiedlicher Stakeholder abzustimmen. Ziel ist es, durch erhöhte Verständlichkeit die Akzeptanz und das Vertrauen in die Modelle zu stärken. Zur Demonstration dient der zweite Anwendungsfall, die Klassifikation von EKG-Signalen, worin unterschiedliche Studien durchgeführt werden, um Erklärungen für verschiedene Personen zu entwickeln und zu evaluieren.

Schließlich zeigt diese Dissertation, dass der vorgeschlagene Entwicklungslebenszyklus und die Methodik für AL-Projekte das Potenzial haben, sowohl die Transparenz als auch die Zusammenarbeit bei der Entwicklung vertrauenswürdiger KI-Systeme erheblich zu verbessern. Das eingeführte LIFEDATA-Framework mit seinem modularen Aufbau und dem Schwerpunkt auf der Rückverfolgbarkeit von Artefakten erweist sich als praktikable Lösung für die Implementierung von AL in verschiedenen Anwendungsbereichen, wie die erfolgreiche Anwendung in den Lebenswissenschaften belegt. Darüber hinaus bietet die domänenspezifische Implementierung einer Methode für interpretierbares ML einen innovativen Ansatz, der dazu beiträgt, die Ausgaben eines KI-Systems auf eine für Menschen verständliche Weise erklärbar zu machen. Im Zusammenhang mit der Integration von Stakeholder-Perspektiven durch den XAI-Compass zeigen die durchgeführten Studien, dass die Einbindung verschiedener Stakeholder die Akzeptanz von Modellinterpretationen verbessert und dabei hilft, diese verständlicher und relevanter für unterschiedliche Nutzergruppen zu gestalten.

Insgesamt liefert diese Dissertation somit neue Ansätze und praktische Lösungen für die Herausforderungen bei der Entwicklung vertrauenswürdiger KI-Systeme mit AL, indem sie Transparenz, Nachvollziehbarkeit und Erklärbarkeit in den Mittelpunkt stellt.

# Acknowledgements

Writing a dissertation is an elaborate and highly intense process that would not have been possible over the past years without the professional and emotional support of several dear people. I want to take this opportunity to acknowledge and express my deepest appreciation to them.

First and foremost, I am profoundly thankful to my doctoral advisor, Prof. Dr. Bernhard Bauer, for his invaluable guidance. Bernhard, your mentorship has been instrumental in navigating my academic journey, and your motivating words have inspired my growth, both professionally and personally - Thank you! I would also like to extend my thanks to Prof. Dr. Jörg Hähner as well as Prof. Dr. Matthias Tichy for kindly agreeing to serve as reviewers of this thesis.

I want to express my gratitude to my close colleagues from the Software Methodology for Distributed Systems group (aka Team SMDS) for their camaraderie, professional discussions, and for enduring my bad jokes. My appreciation further goes to the numerous students whose dedication to their theses and projects has played a crucial role in helping me achieve my goals. A special mention goes to Benjamin Weigell - Benni, your support as a student assistant and now as a colleague has been indispensable. I eagerly anticipate our future adventures, and may we never run out of cappuccino or ice cream!

Furthermore, I want to acknowledge the members of the Center for Responsible AI Technologies. Working with all of you over the past few years has been a privilege. I am particularly thankful for the inspiring collaboration with Prof. Dr. Kerstin Schlögel-Flierl and Paula Ziethmann: Kerstin and Paula, it is a pleasure to work alongside you!

I am especially appreciative of Team LIFEDATA for the memorable time we shared during the project phase. My sincere thanks go to my industry research partner Corpuls, including Marius Nann, Sarah Dopfer, and Anton Roth, as well as the LIFEDATA contributors Peter Kienle, Gregor Mühlegger, Lena Essig, and Jens Röser. Your excellent cooperation and the many moments of shared curiosity, occasional bewilderment, surprise, and success made this journey truly special. A heartfelt thank you goes to my former colleague, namesake, and MLOps wizard, Fabian Rabe. Fabi, your patience, strength, and unwavering support have been invaluable. I could always count on you when I needed it. You are not only Fabi the 1st but also the true AI hero!

When reflecting on this journey, I want to express my gratitude to those with whom I haven't been able to spend as much time as they deserved over the past years but who have always stood by me. I thank all my cherished friends, especially Nico, Michi and Bene, for their unwavering support from the very beginning.

Finally, and this is particularly important, I owe a tremendous debt of gratitude to my parents. Their decades of unwavering support have made it possible for me to pursue my path. Mum and Dad, your help has been invaluable!

Last but definitely not least, I express my deepest gratitude to Karo, who stood by me throughout the entire duration of my doctorate and thus has the greatest share in this dissertation. She is the most lovable and warm-hearted person I could imagine. Karo, your unwavering belief in me and constant support and confidence gave me the strength I needed. THANK YOU so much for your boundless encouragement and unconditional love!

I am glad to have such wonderful people in my life! ♡

*– Yours, Fabi.*

# Contents

**Abstract** iii

**Zusammenfassung** v

**Acknowledgements** vii

**Contents** ix

**I. Introduction and Foundations** 1

**1 Introduction** 3

1.1 Motivation 3

1.2 Challenges 4

1.2.1 Development Process in Active Learning Projects 4

1.2.2 Design of Traceability-Aware Active Learning Systems 5

1.2.3 Tailored Explainability in Trustworthy AI Systems 6

1.3 Objectives and Contributions 7

1.3.1 Active Learning Development Life Cycle and Engineering Methodology 7

1.3.2 Framework for Traceable Active Learning Projects 8

1.3.3 Integration of Interpretable Machine Learning 9

1.4 Outline 10

1.5 Scientific Participation 13

1.5.1 Research Projects 13

1.5.2 Supervised Theses 16

1.5.3 Publications 18

**2 Foundations** 23

2.1 Machine Learning 24

2.1.1 Learning Types 24

2.1.2 Artificial Neural Network 26

2.1.3 Active Learning 31

2.2 Trustworthy Artificial Intelligence 38

2.2.1 Ethical Principles 39

2.2.2 Key Requirements 40

2.3 Explainable Artificial Intelligence 43

2.3.1 Interpretable Machine Learning 43

2.3.2 Methods for Local Post-hoc Explanations 46

2.4 Software Engineering for AI Systems 49

2.4.1 Process Models for Data-intensive Systems 49

2.4.2 Operation Frameworks 51

## II. Development in AL Projects<br>A Process Perspective 55

**3 Active Learning Development Life Cycle** **57**
3.1 Project Process Model . . . 58
3.1.1 Scoping & Project Planning . . . 58
3.1.2 Data Engineering . . . 59
3.1.3 Modeling . . . 60
3.1.4 Maintenance . . . 60
3.1.5 Acceptance . . . 61
3.2 Inception . . . 62
3.2.1 Cold Start . . . 62
3.2.2 Warm Start . . . 63
3.3 Continuous Phases . . . 65
3.3.1 Data Iteration . . . 65
3.3.2 Machine Learning Iteration . . . 68
3.3.3 Development Iteration . . . 73
3.3.4 Operations Phase . . . 76
3.4 Roles . . . 81
3.4.1 Non-technical Roles . . . 81
3.4.2 Technical Roles . . . 81
3.5 Summary . . . 83

**4 Engineering Methodology** **85**
4.1 Developing in Data-Centric Projects . . . 86
4.2 Basic Concepts . . . 86
4.2.1 Data Principle . . . 87
4.2.2 Code Principle . . . 89
4.2.3 Automation Principle . . . 90
4.3 Branching Workflow . . . 92
4.3.1 Main Branch . . . 92
4.3.2 Feature Branches . . . 93
4.3.3 Experiment Branches . . . 94
4.3.4 Release Branch . . . 94
4.3.5 Production Environment . . . 95
4.4 Git Workflow for Active Learning . . . 96
4.5 Evaluation . . . 99
4.5.1 Literature Based Approach . . . 99
4.5.2 Expert Interviews . . . 110
4.5.3 Metadata Analysis . . . 116
4.6 Summary . . . 120

## III. Design of AL Projects<br>A System Perspective 121

**5 Framework Design Approach** **123**
5.1 Related Work . . . 124
5.2 Overall Process . . . 127
5.3 Core Framework . . . 128
5.3.1 User Management and Interfaces . . . 128

5.3.2 Annotation Domain Layer . . . . . 131
5.3.3 Data Persistence Layer . . . . . 133
5.3.4 Project Persistence Layer . . . . . 136
5.4 Project Instance . . . . . 137
5.4.1 Basic MLOps Components . . . . . 137
5.4.2 Data Import . . . . . 138
5.4.3 Machine Learning Service . . . . . 139
5.4.4 Webinterface Components . . . . . 143
5.5 Limitations . . . . . 144
5.6 Summary . . . . . 146

**6 Use Cases & Evaluation** **147**
6.1 Skin Image Analysis . . . . . 148
6.1.1 Data Basis . . . . . 148
6.1.2 Application . . . . . 151
6.1.3 Active Learning Simulation . . . . . 157
6.2 ECG Signal Classification . . . . . 161
6.2.1 Data Basis . . . . . 161
6.2.2 Application . . . . . 164
6.2.3 Active Learning Simulation . . . . . 174
6.2.4 Human-in-the-loop Study . . . . . 179
6.3 Summary . . . . . 182

**IV. Interpretable Machine Learning An Integration Approach** **183**

**7 Domain Specific Model Interpretations** **185**
7.1 Interpretable Machine Learning in Skin Image Analysis . . . . . 186
7.2 Dermatologist's Human Approach . . . . . 188
7.3 Explainer for Skin Image Classifier . . . . . 189
7.3.1 Perturbation Dimensions . . . . . 190
7.3.2 Implementation . . . . . 191
7.4 Application . . . . . 193
7.4.1 ML Pipeline Adaption and Experimental Setup . . . . . 193
7.4.2 Hypothesis . . . . . 194
7.5 Empirical Results . . . . . 195
7.5.1 True Positives . . . . . 195
7.5.2 False Negatives . . . . . 197
7.5.3 Aggregation . . . . . 198
7.6 Discussion . . . . . 200
7.7 Summary . . . . . 201

**8 Stakeholder-oriented Model Interpretations** **203**
8.1 XAI-Compass: A Conceptual Guide for Stakeholder Engagement . . . . . 204
8.1.1 Goals of Model Interpretations . . . . . 205
8.1.2 Model Interpretations across the Life Cycle . . . . . 206
8.1.3 Stakeholder Groups of Model Interpretations . . . . . 207
8.1.4 Touchpoints for Model Interpretations . . . . . 209
8.2 Case Study . . . . . 210
8.2.1 Interpretable Machine Learning for ECG Signal Classification . . . . . 210

8.2.2 Implementation . . . 212
8.3 Stakeholder Engagement . . . 215
8.3.1 Domain Expert Interviews . . . 215
8.3.2 Scenario-based Focus Group Discussions . . . 218
8.3.3 Survey at Workshop . . . 222
8.3.4 Discussion and Recap . . . 225
8.4 Summary . . . 226

**V. Conclusion and Outlook** **227**

**9 Conclusion and Outlook** **229**
9.1 Development in AL Projects . . . 229
9.2 Design of AL Systems . . . 230
9.3 Interpretable Machine Learning . . . 231
9.4 Future Work . . . 232

**Bibliography** **235**

**List of Abbreviations** **257**

# List of Figures

1.1 Thesis outline . . . . . 10

2.1 Research fields and their relevance to the thesis' parts . . . . . 23
2.2 Schematic representation of a Multi-layer Neural Network . . . . . 29
2.3 Visual depiction of a convolutional neural network . . . . . 30
2.4 Three different Active learning scenarios . . . . . 32
2.5 Pool-based Active learning cycle . . . . . 33
2.6 Schematic visualization of different query strategies . . . . . 34
2.7 Taxonomy of query strategies . . . . . 34
2.8 High-level framework for trustworthy AI . . . . . 39
2.9 Simplified taxonomy of IML methods . . . . . 44
2.10 Comparison of process models for data-intensive systems. . . . . . 50

3.1 Active learning development process . . . . . 58
3.2 Potential stages of inception . . . . . 62

4.1 Development dataset concept . . . . . 88
4.2 Proposed project structure . . . . . 89
4.3 Concept of a minimal infrastructure setup . . . . . 91
4.4 Git workflow with Experiment-Branches . . . . . 93
4.5 Git workflow with production environment . . . . . 96
4.6 Git workflow for an active learning project . . . . . 97
4.7 Evaluation concept for the engineering methodology proposal . . . . . 99
4.8 Project maturity levels along the dimensions of people, process and technology . . . . . 112
4.9 Overview of runner execution metadata in the skin image analysis use case project . . . . . 118
4.10 Overview of runner execution metadata in the ECG signal classifier use case project . . . . . 118

5.1 Process outline in an active learning project with LIFEDATA . . . . . 127
5.2 Reference architecture of active learning projects using LIFEDATA . . . . . 128
5.3 Class diagram of events . . . . . 132
5.4 Initial database schema in LIFEDATA . . . . . 135
5.5 Initial ML pipeline definition in an LIFEDATA project instance . . . . . 140

6.1 Examples of lesion images from the HAM10000 dataset . . . . . 149
6.2 Class- and lesion distribution of samples . . . . . 150
6.3 Sample distribution in relation to patient population . . . . . 150
6.4 Project setup for the skin image use case . . . . . 151
6.5 ML pipeline definition of the LIFEDATA project instance applied to the skin analysis use case . 153
6.6 Fitting process of the skin image classifier during model training . . . . . 154
6.7 Confusion matrix of the trained skin image classifier . . . . . 156
6.8 Average classification performance of the skin image classifier . . . . . 157
6.9 Skin image classification performance per class . . . . . 158
6.10 Query set proportion per class . . . . . 160
6.11 Overview of sample and label frequencies . . . . . 162
6.12 Label co-occurrence matrix . . . . . 163
6.13 Distribution of samples in relation to patient population . . . . . 164
6.14 Project setup for the ECG signal classification use case . . . . . 165

6.15 LIFEDATA’s ML pipeline definition applied to the ECG signal classification use case . . . . . . 168
6.16 Fitting process of the ECG signal classifier during model training . . . . . . . . . . . . . . . . . . 170
6.17 Confusion matrix of the trained ECG signal classifier . . . . . . . . . . . . . . . . . . . . . . . . . 171
6.18 Classification performance of class ′SNR′ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 175
6.19 ECG signal classification performance per class . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
6.20 Query set proportion of class ′SNR′ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177
6.21 Query set proportion per class . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178
6.22 Implementation of the inter- and intra-annotator agreement study . . . . . . . . . . . . . . . . . 179

7.1 Local model output interpretations of the DNN-based classifier . . . . . . . . . . . . . . . . . . . 187
7.2 Perturbation dimensions of the explainer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 190
7.3 Overview of the observed model behavior for two correctly classified samples . . . . . . . . . . . 196
7.4 Overview of the observed model behavior for two incorrectly classified samples . . . . . . . . . . 197
7.5 Aggregated model predictions for nevus and melanoma . . . . . . . . . . . . . . . . . . . . . . . 198

8.1 XAI-Compass - A schematic overview of stakeholder engagement in XAI . . . . . . . . . . . . . 204
8.2 Perturbation dimensions for ECG signal data for sampling for local exploration in LIME . . . . 212
8.3 12-lead ECG signal with highlighted shapley values at sample-level . . . . . . . . . . . . . . . . 213
8.4 Lead-wise Shapley values for 12-lead ECG signals . . . . . . . . . . . . . . . . . . . . . . . . . . 214
8.5 Comparative visualization techniques for local ECG classification model interpretations . . . . 216
8.6 Trust in AI generated output across escalation scenarios . . . . . . . . . . . . . . . . . . . . . . 223
8.7 Survey result about clarity of provided explanations . . . . . . . . . . . . . . . . . . . . . . . . . 224

# List of Tables

4.1 Aggregated representation of best practices and their fulfillment by the concepts presented . . 109
4.2 Profiles of interviewees . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 110
4.3 Inventory of project maturity levels and self-assessment of interviewees . . . . . . . . . . . . . . 112

5.1 AL framework comparison . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
5.2 Overview of available commands integrated in LIFEDATA's CLI . . . . . . . . . . . . . . . . . . 129
5.3 Overview of REST-API methods by layer provided in LIFEDATA . . . . . . . . . . . . . . . . . . 131

6.1 Class description of lesions within the HAM10000 dataset . . . . . . . . . . . . . . . . . . . . . 149
6.2 Provided metadata within the HAM10000 dataset . . . . . . . . . . . . . . . . . . . . . . . . . . 149
6.3 Aggregated model performance metrics of the skin image classifier . . . . . . . . . . . . . . . . 156
6.4 Relevant data sources with ECG signal data . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
6.5 Available metadata of the ECG records . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
6.6 Class description of observed 12-lead ECG signal samples . . . . . . . . . . . . . . . . . . . . . 163
6.7 Classification report of the ECG signal classifier . . . . . . . . . . . . . . . . . . . . . . . . . 172
6.8 Aggregated model performance metrics of the ECG signal classifier . . . . . . . . . . . . . . . . 172
6.9 Inter-annotator-agreement of the four annotators . . . . . . . . . . . . . . . . . . . . . . . . . 181
6.10 Self-agreement of the four annotators . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 181

7.1 Evaluation results of the trained skin image classifier based on the MobileNetV2 architecture . 193

8.1 Overview of participants in the scenario-based focus group discussions . . . . . . . . . . . . . . 219
8.2 Comparative summary of feedback from Model Consumers and Model Builders . . . . . . . . . 221

# List of Listings

5.1 Code snippet of the CLI logic for initializing a project using the project template . . . . . . . 130
5.2 Defintion of the dataclass „Assignment“ in Python. . . . . . . . . . . . . . . . . . . . . . . . . . . 132
5.3 Excerpts from the „Assignment Service “ implementation . . . . . . . . . . . . . . . . . . . . . 134
5.4 Definition of the „Assignment“ data model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
5.5 Extended „Assignment Repository“ implementation . . . . . . . . . . . . . . . . . . . . . . . 136
5.6 Example configuration of the query stage . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141
5.7 Query stage in ML pipeline definition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142

# I. Introduction and Foundations

# Introduction 1

**1.1 Motivation . . . . . . . . . 3**
**1.2 Challenges . . . . . . . . . 4**
1.2.1 Development Process in Active Learning Projects . 4
1.2.2 Design of Traceability-Aware Active Learning Systems . . . . . . . . . . . 5
1.2.3 Tailored Explainability in Trustworthy AI Systems . 6
**1.3 Objectives and Contributions . . . . . . . . . . . . . 7**
1.3.1 Active Learning Development Life Cycle and Engineering Methodology 7
1.3.2 Framework for Traceable Active Learning Projects . 8
1.3.3 Integration of Interpretable Machine Learning . . . . . 9
**1.4 Outline . . . . . . . . . . . 10**
**1.5 Scientific Participation . . 13**
1.5.1 Research Projects . . . . . 13
1.5.2 Supervised Theses . . . . . 16
1.5.3 Publications . . . . . . . . . 18

*„We can only see a short distance ahead,*
*but we can see plenty there that needs to be done.“*
— Alan Mathison Turing 1950

With this quote from his paper on computing machinery and intelligence, where he first introduced concepts of the famous Turing Test, a method proposed to evaluate a machine's ability to exhibit intelligent behavior indistinguishable from that of a human, Alan Turing highlights the ongoing challenges and opportunities in the development of artificial intelligence (AI). Although this statement, whose author regarded as a pioneer of computer science and AI, dates back more than 70 years, the words still hold enduring significance.

The following chapter motivates the topic of this dissertation, highlighting three current challenges explored in this context as well as the resulting objectives of this thesis. In addition to an overview of the content of this dissertation, the relevant contributions made during its preparation are likewise highlighted.

## 1.1 Motivation

Today, the attention on AI technology is unprecedented (Van Noorden *et al.* 2023). In a constantly evolving world, it's becoming increasingly clear that AI is permeating far-reaching aspects of daily life, with its impact felt across various sectors, including healthcare, finance, education, and beyond (Maslej *et al.* 2023).

The potential of AI has long moved beyond the purely academic realm and has become a subject of societal discourse. As AI systems become more deeply embedded in society, the need for these systems to be functionally intelligent and trustworthy is emphasized. Thus, pursuing trustworthy AI is no longer solely a scientific endeavor but a societal imperative aimed at aligning technological progress with ethical standards and public expectations (Brundage *et al.* 2020; Kaur *et al.* 2023; Li *et al.* 2023). In this context, the European Commission's High-Level Expert Group on AI published in 2019 a framework for trustworthy AI, defining four ethical principles as foundational, along with seven key requirements: human agency and oversight, technical robustness and safety, privacy and data governance, transparency, diversity, non-discrimination and fairness, environmental and societal well-being, as well as accountability (High-Level Expert Group on AI 2019).

In light of the need to implement AI systems trustworthy, the focus on how these systems are developed becomes paramount. At the core of most AI systems lie algorithms; statistical models trained through machine learning (ML) methods to make predictions or decisions based on data, or to generate new outputs in a way that appears intelligent to a human (Russell *et al.* 2016). The primary difference between ML and traditional algorithms is that ML derives the knowledge needed to solve a task through stochastic strategies directly or extract them from the available data, rather than being defined through the explicit programming of individual instructions (James *et al.* 2013).

Among the various training paradigms for ML models, supervised learning, which is conducted based on sample data, is notable for its reliance on labeled training data. This reliance uncovers another significant issue: the data labeling process is labor-intensive and one of the most expensive steps in developing supervised learning models (Fredriksson *et al.* 2020b).

Human-in-the-loop ML methods have emerged as promising approaches in this context, integrating human expertise into the training process (Mosqueira-Rey *et al.* 2023). This combination improves the model's learning process and supports aspects of trustworthy AI by incorporating human feedback directly into the AI system's learning cycle. Active learning (AL), a specific form of ML that involves human participation, exemplifies this by allowing the learning system to consult a human oracle to label unknown data points, thereby iteratively improving the model's performance (Settles 2009).

Through this approach, AL addresses two distinct demands: optimizing the labeling process and advancing the development of trustworthy AI systems. Motivated by the potential that AL can enhance the way we approach both the efficiency of data labeling and the overarching goal of developing trustworthy AI systems, this dissertation aims to explore the synergies between AL and the concepts of trustworthy AI with a focus on the requirement for transparency, which is closely linked to the ethical principle of explainability (High-Level Expert Group on AI 2019). Implementing transparency and explainability within trustworhty AI presents not only technical but also conceptual challenges to the processes and systems, which are detailed in the following section.

## 1.2 Challenges

The following section presents three main problems that arise when implementing and integrating the AL methodology into trustworthy AI systems. These problems are briefly introduced to provide an overview of the challenges of this dissertation and will be discussed in further detail in the respective parts.

### 1.2.1 Development Process in Active Learning Projects

The development of AI projects that implement the AL methodology confronts development teams with a number of problems (Arpteg *et al.* 2018). As they are data-centric in nature, these are characterized by the rapid evolution of the data landscape, the high resource requirements for creating and managing training data (Sculley *et al.* 2015), and the methodological heterogeneity within the teams (Diaz-de-Arcaya *et al.* 2023). There is a consensus in the research community, that traditional process models need to be revised (Haakman *et al.* 2021; Machado *et al.* 2024), not only to meet the specific requirements of integrating an AL loop into a project. Further requirements in this context are particularly relevant to trustworthy AI, where the traceability of each asset throughout the entire life cycle plays an essential role (Habibullah *et al.* 2023).

Typically, teams comprise a variety of professionals whose methodological approaches and perspectives on a project can vary significantly. For example, software developers usually bring a strong familiarity with software engineering (SE) concepts into projects, encompassing a systematic, structured approach. They tend to view processes and problem-solving through the lens of established development practices, where the traceability of artifacts is a well-established practice.(Martínez-Fernández *et al.* 2022)

On the other hand, data scientists adopt a more experimental working method, with a strong focus on iteratively improving algorithms through experimentation and learning (Makinen *et al.* 2021). This approach is essential in developing trustworthy AI but can lead to a less structured workflow that is more difficult to integrate into traditional software development cycles (Rule *et al.* 2020). The challenge of traceability becomes particularly evident here, as data scientists' iterative development cycles and experimental approaches require comprehensive documentation and tracking of all changes and decisions to ensure transparency and comprehensibility.

The coexistence of these different working styles in a team, without a clear process that allows for the integration and coordination of different approaches, can impair development efficiency and hinder the fulfillment of requirements for trustworthy AI. The shortcomings of traditional process models exacerbate this: Established approaches are either too rigid and sequential to effectively respond to the emerging changes in data within an AL project. They often do not offer the necessary flexibility required to deal with the complexity and dynamics of AL projects.

**Challenge I**

*Established process models for AI projects need to be revised, especially when integrating an AL loop. Heterogeneous engineering teams need a methodology to manage the dynamic data to implement data-centric AI projects effectively and harmonize different development approaches.*

### 1.2.2 Design of Traceability-Aware Active Learning Systems

The development of trustworthy AI necessitates a traceability system that spans from the initial data acquisition to the outcomes of the trained model, including user inputs and outputs generated by the ML model. One of the key requirements in the realisation of trustworthy AI results in the challenge of achieving end-to-end traceability (High-Level Expert Group on AI 2019), extending from the initial data origin to the final model output. This level of traceability goes beyond the artifacts produced throughout the life cycle. In an AL system, this includes both user inputs via given annotations and the outputs generated by the model in the form of predictions. Achieving this level of traceability contributes further to reproducibility and transparency in developing trustworthy AI systems and enhances the ability to explain system behavior (Pineau *et al.* 2019; Samuel *et al.* 2021).

From a system perspective, building an AL loop that adheres to the requirements of trustworthy AI presents developers with the challenge of integrating numerous technical components like a workflow orchestration component (Kreuzberger *et al.* 2023; Steidl *et al.* 2023). The integration of these components becomes particularly necessary as AI systems transition from a laboratory setting into operational deployment, requiring robust frameworks that can support continuous training systems. This includes, for instance, the implementation of feedback loops and the integration of components for the seamless logging of data origins as well as the continuous versioning of all artifacts during the execution of the ML pipeline in a continuous training system (Isdahl *et al.* 2019).

Traditional frameworks for implementing AL systems prove to be inadequate as they are primarily focused on executing the AL methodology and neglect traceability, which is relevant for trustworthy AI. This gap in existing approaches underscores the need for a framework that implements both dimensions: the functionalities of the AL methodology for effective data annotation and the integration of data provenance and artifact versioning technologies.

**Challenge II**

*Frameworks for AL projects do not cover the requirements for End-to-End traceability in AL systems. About trustworthy AI development, a framework is required that covers the AL functionality and the technical components, including the data provenance of user input in the form of annotations and the versioning of all artifacts created in the AL system.*

### 1.2.3 Tailored Explainability in Trustworthy AI Systems

In the quest to develop trustworthy AI systems, the challenge of explainability is multifaceted and touches upon the need for AI systems to be transparent in both their technical processes and the human decisions that guide their application. This means that such systems must be capable of providing insights into their decision-making processes in a manner understandable to humans.(High-Level Expert Group on AI 2019)

Complicating the landscape is the trend toward using increasingly complex models to process growing volumes of data within data-centric AI systems. These models, such as deep neural networks (DNNs), are powerful for many problem definitions but often reach a level of complexity where their decision-making processes and inner workings become difficult for humans to explain, leading to their common characterization as Black-Boxes. This complexity of models used in AI systems poses a significant hurdle on the path to technical explainability. The potential ambiguity of outputs generated by these models is exemplified by phenomena like the „Clever Hans Effect“, where a model's seemingly correct predictions are based on mistaken correlations in the training data (Lapuschkin *et al.* 2019). Such instances highlight the importance of interpretable machine learning (IML), aiming to make the decision-making process of AI systems transparent, ensuring that their operations are functionally correct and understandable.

However, the pursuit of enhanced technical explainability may lead to compromises, often due to the fact that increasing a system's interpretability can come at the cost of its accuracy and vice versa (High-Level Expert Group on AI 2019). This dilemma stems from technological considerations as well as from the usability of the systems (Bell *et al.* 2022), where, especially in high-risk scenarios, different stakeholders must be able to demand comprehensive explanations of how an AI system arrives at its decisions (High-Level Expert Group on AI 2019).

Therefore, explanations must be tailored to the expertise level of the concerned stakeholders (Miller 2019; Laato *et al.* 2022). This necessitates the domain-specific implementation of IML methods and, on the other hand, an adaptable level of detail in explanations to ensure they are accessible to different stakeholders (Longo *et al.* 2024).

**Challenge III**

*Addressing the layered challenge of (technical) explainability in developing trustworthy AI requires an integrative approach. Overcoming the complexity of AI systems necessitates applying interpretable machine learning methods, their customization to the domain, and providing stakeholder-specific explanations.*

## 1.3 Objectives and Contributions

While the previous section presents the challenges addressed in this dissertation, the following section is dedicated to the objectives to solving these challenges.

### 1.3.1 Active Learning Development Life Cycle and Engineering Methodology

As outlined in Section 1.2.1, developers face numerous hurdles when implementing data-centric AI projects, especially those employing an AL loop. These issues are associated with the dynamic nature of data and the requirement for trustworthy AI to track artifacts throughout the life cycle. Additionally, the diverse methodological approaches prevalent within various development teams add complexity.

To adapt traditional process models and development flows to this increasing complexity, this dissertation proposes a revised AL development life cycle model and an innovative development methodology for AL projects.

**Objective 1**

*Provide an AL development life cycle model and an engineering workflow tailored for AL projects. The primary goal is to establish a unified methodological foundation that enhances the efficiency and transparency of implementing AL projects. This approach aims to facilitate the management of challenges such as data volatility, high resource requirements for artifact creation, and methodological diversity within teams.*

**Contributions:**

- **AL Development Process Model:** A detailed schema that structures the development process into project phases and individual steps, integrating established development concepts. It guides engineering teams through the AL development life cycle, defining specific activities and roles, thereby offering clear structure and direction.
- **Coordinated Development Methodology:** A step-by-step guide derived from the process model to support teams in the practical execution of projects. This includes concepts related to code, data, and automation like continuous integration, and deployment pipelines, along with an innovative branching workflow. The development methodology focuses on effectively integrating various approaches, ensuring efficient project realization.

### 1.3.2 Framework for Traceable Active Learning Projects

As described in Section 1.2.2, the development of trustworthy AI necessitates a comprehensive traceability system. This system must track every stage of the process, from the initial acquisition and preparation of data to the final outputs of the trained model, while also capturing user interactions and the model-generated predictions. This contributes to transparency and reproducibility - from a systemic perspective, implementing AL projects that aim to meet these requirements poses a significant issue for developers in integrating numerous components.

In order to provide development teams with a blueprint that supports them in the implementation of AL projects, this dissertation presents an innovative framework that includes both the functionalities required for effective data annotation within the AL methodology and the integration of essential technologies for data provenance and artifact versioning.

**Objective 2**

*Development of a framework that equips AL systems with the essential components for end-to-end traceability. This includes ensuring the data provenance of user inputs in the form of annotations and facilitating the versioning of all artifacts generated within the AL life cycle.*

**Contributions:**

- **Open-Source Framework:** Introduction of a open-source framework that enables teams to carry out AL projects effectively. It establishes a foundation for end-to-end traceability and outlines a comprehensive approach for the technical implementation of the required components.
- **Generic Project Template:** Provision of a reusable and adaptable project template as an integral part of the framework which includes key components essential for realizing AL projects with a focus on traceability.
- **Application in two Case Studies:** By demonstrating the utility of the framework through two use cases in the life sciences domain, this dissertation further contributes to the application of complex ML pipelines in the classification of skin images and ECG signals, as well as the evaluation of different QSs in these scenarios.

### 1.3.3 Integration of Interpretable Machine Learning

The challenges regarding explainability in developing trustworthy AI systems mentioned in Section 1.2.3 involve the necessity for complex technical procedures utilized in AI systems to be explained in a human-comprehensible form to diverse stakeholders.

To address the adaptation of technical explanations of AI systems to both the domain and user, this dissertation proposes an approach for integrative approach aiming to involve different stakeholders in the development process of trustworthy AI.

**Objective 3**

*Enhance the accessibility of AI explanations by developing an interpretation algorithm supplemented with domain knowledge to increase the relevance of model interpretations for a non-technical audience within a selected domain and, furthermore, adapting model interpretations to specific stakeholder groups to increase the relevance of AI explanations for the respective users.*

**Contributions:**

- **Domain-Specific Interpretation Algorithm:** Implementing an approach tuned for an AI-based skin image classifier to elucidate the model's outputs.
- **Stakeholder Integration Framework:** Provision of a visual tool that systematically aligns stakeholder groups, life cycle stages, objectives and possible touchpoints for model interpretations.
- **Interdisciplinary Studies:** Execution of studies employing social science methods to capture a broad spectrum of stakeholder perspectives on model interpretations for an AI-based ECG signal classifier.

## 1.4 Outline

This section provides an overview of the structure of this dissertation. Figure 1.1 provides the thesis outline, illustrating the chapter division's general structure and thematic interrelations.

Following the introductory part in Chapter 1, Chapter 2 present the foundations associated with this work. The thesis's three main parts II, III, and IV are each divided into two chapters. The structure of the chapters is designed such that the reader should follow the sequence as the content build upon each other.

Part II, encompassing Chapters 3 and 4, introduces the development concepts in AL projects from a process perspective. Part III, comprising Chapters 5 and 6, focuses on a systems perspective in the design of AL projects. The latter chapter further introduces the two case studies that extend to the chapters in the following section. Subsequently, Part IV describes a stakeholder integration approach, with Chapter 7 addressing the domain perspective and Chapter 8 discussing studies involving various stakeholders in an AI project. The dissertation concludes with a summary in Chapter 9, which offers a perspective on future work.

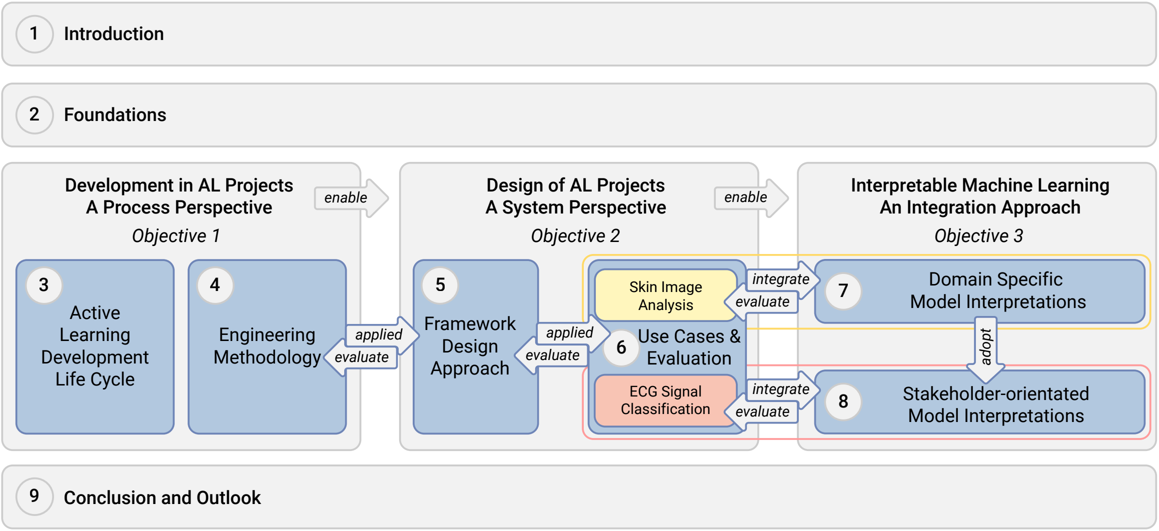


**Figure 1.1:** Thesis outline. The overall structure of this dissertation shows the chapters organized in three main parts, as well as their interrelationships and the integration of two use cases.

## Part I. Introduction and Foundations

**Chapter 1: Introduction.** The thesis begins by describing the current progress in the field of trustworthy AI and the challenges associated with its development. Based on this motivation, the objectives and the resulting approaches are presented. Furthermore, the research contributions are presented. This summarizes the publications that already contain parts of this dissertation, as well as the related research projects.

**Chapter 2: Foundations.** This chapter introduces the basic aspects of developing AI systems. Initially, an introduction to the technical concepts of AI and ML is provided, followed by an overview of AL and its associated foundations. This includes both AL scenarios and a review of established strategies for querying samples, which are an integral part of the methodology. The chapter then delves into the field of trustworthy AI, focusing on the technologically relevant concepts for this dissertation, such as IML methods. The chapter closes with a brief overview of the fundamentals of SE pertinent to the development of AI systems.

## Part II: Development in Active Learning Projects<br>A Process Perspective

**Chapter 3: Active Learning Development Life Cycle.** This chapter presents the proposal of a development life cycle model for AL projects, which is based on the principles of traditional process models and the agile concepts of DataOps, MLOps, and DevOps and combines them. The aim is to synchronize the established practices of the individual phases, iterations, and their relationships to develop AL projects.

**Chapter 4: Engineering Methodology.** In this chapter, the development life cycle model for AL projects from Chapter 3 is taken up and translated into a methodology that enables interdisciplinary teams to synchronize the implementation tasks which are performed in the individual phases and iterations of the AL project. To this end, practicable guidelines are presented to enable engineering teams to overcome the challenges that arise during implementation (**Objective 1**). The concepts presented are then evaluated in a three-stage process.

## Part III. Design of Active Learning Projects<br>A System Perspective

**Chapter 5: Framework Design Approach.** This chapter introduces LIFEDATA, an open-source framework that can be used for the development of AL projects. It addresses **Objective 2** by providing developers with a tool that can be used as a starting aid in realizing AL projects, whereby the implemented concepts promote traceability.

**Chapter 6: Use Cases & Evaluation.** The framework described in Chapter 5 is applied in two use cases in Chapter 6, which serve to evaluate the implemented concepts. While simulations are carried out in skin image analysis, the use case of ECG signal classification depicts a real scenario with human-in-the-loop.

## Part IV. Interpretable Machine Learning An Integration Approach

**Chapter 7: Domain Specific Model Interpretations.** Part IV of this dissertation addresses the requirement of explainability of trustworthy AI. To cover **Objective 3**, Chapter 7 presents a domain-specific approach that was developed specifically for the use case of skin image analysis. Here, a human-understandable approach for classification of skin leasons is combined with an ML model interpreation algorithm. This approach is then presented and analyzed using selected examples.

**Chapter 8: Stakeholder-orientated Model Interpretations.** This chapter introduces the XAI-Compass, a framework that can be used to consider the different perspectives on model interpretations in an AI system. For this purpose, conceptual solutions are presented on how the different touchpoints can be technically implemented, which are concretized using the ECG signal classification use case. Furthermore, three studies will be carried out.

## Part V. Conclusion and Outlook

**Chapter 9: Conclusion and Outlook.** The final chapter of this dissertation provides a summary of all the concepts and results discussed throughout. In this chapter, the objectives defined at the outset of the thesis are recapitulated. Additionally, the results are critically reflected upon to ascertain their scientific and practical significance. Furthermore, the chapter offers an outlook on potential future research directions inspired by the findings of this thesis.

## 1.5 Scientific Participation

While working on this dissertation, participation in various projects and research associations took place. These collaborations provided insights into the practical applications and interdisciplinary implications of the research related to this dissertation. In addition to the project collaborations, this section discusses supervised theses and published papers directly associated with this dissertation.

### 1.5.1 Research Projects

The following section provides a concise overview of research projects and their themes, elucidating the overarching objectives related to this dissertation. In addition to presenting the projects, a separate focus is placed on the author's personal contributions to each project.

#### Life Sciences Improved by a Framework for Efficient Data Annotation Through Active Learning (LIFEDATA)

**Summary:** Over recent years, the surge in medical digitization has amassed a wealth of data. A significant portion of this data aids in diagnosing diseases. Using existing annotated data, ML methods can be used to train models that make it possible to make predictions on new data. These technologies are aimed at applications such as the use of AI-based clinical decision support systems. Nevertheless, making accurate predictions, especially using deep learning (DL) models, necessitates a substantial foundation of annotated data that adequately captures even rare diagnoses. The procurement of such a dataset is either resource-intensive or leads to underutilized extant data.

In response to this challenge, an open-source framework was developed within the LIFEDATA project. The eponymous framework melds AL methodologies with Deep Neural Networks. Its sophisticated design autonomously identifies and prioritizes high-value data points, such as rare diagnoses, for expert annotation. This streamlined approach promotes efficient ML model training. Through the integration of semi-supervised learning in combination with AL, a significant portion of annotations can be generated with minimal manual intervention.

The collaborative project, spanning 3.5 years, was funded by the German Federal Ministry of Education and Research (BMBF)[1]. Collaborators included the University of Augsburg, the GS Elektromedizinische Geräte G. Stemple GmbH, Kaufering (Corpuls), and the German Heart Center of the Technical University of Munich. Through the lens of two distinct Life Sciences case studies – skin image analysis and the classification of ECG signals – the initiative evaluated the framework's versatility and applicability across various data challenges. Another key objective was to integrate algorithms that provide the interpretation of model classifications, ensuring the iterative ML model decisions are transparent and explainable.

[1] Reference number 031L9196B

**Personal Contribution:** The project followed a biphasic structure. During the initial phase, the open-source framework was developed and refined based on a previously annotated dataset. During this stage, the foundational architecture was established, and primary functionalities were implemented.

This foundational work paved the way for the integration of the secondary use case in the subsequent phase of the project. Contributions spanned from implementing core functionalities of the primary framework to actively participating in the setup of complex ML pipelines across both use cases. In addition to hands-on engagement with the core framework and ML pipelines, a significant emphasis was placed on conceptualizing and integrating features for semi-supervised ML. Additionally, various algorithms for model interpretation were developed and applied, catering to diverse data types and model architectures.

Post-development, there was participation in the deployment of the application. Involvement extended to the technical realization, with primary responsibility being attributed to the long-term provision and optimization of all requisite components and their underlying infrastructure. Another major contribution was accompanying the conceptualization and technical execution of multi-annotator functionalities. This entailed collaborative implementation as well as planning and scheduling experiments, as well as the realization of a dashboard to analyze the entire project.

**Dissemination:** At the end of the second project phase, the open sourcing process for the relevant software components was prepared and carried out. In addition, the following talks were held:

- **Trustworthy AI in medicine - from efficient data annotation to intuitive model explanation.** The talk, originally titled „Vertrauenswürdige KI in der Medizin - von effizienter Datenannotation bis intuitiver Modellerklärung.", co-presented with Bernhard Bauer and Marius Nann, encompassed an interactive workshop where the project was introduced to a professional audience at the conference *KI-Fabrigk @WIKOIN22* in Ingolstadt.[2] The objective was to educate about trustworthy AI by demonstrating the concepts of transparency and model interpretation implemented in LIFEDATA. During a live poll, workshop participants were able to vote on their level of trust in the developed model and the training data used.
- **LIFEDATA - A Framework for Traceable Active Learning Projects.** A curated selection of scientific contributions was showcased at the First Bavarian International Conference on AI (*AI.BAY 2023*), which took place on February 23 and 24, 2023, at the „Forum der Zukunft"of the German Museum in Munich. As part of a poster session, the LIFEDATA project was presented[3], highlighting the innovation of the framework, its technical implementation, and the results of the two use cases.

[2] Relevant passages are found in this dissertation in Part II, Chapter 6, and in Part IV, Chapter 8

[3] Relevant passages are found in this dissertation in Part III, Chapter 5 and 6

### Center for Responsible AI Technologies

**Summary:** Founded in 2022 by the Technical University of Munich, the University of Augsburg, and the Munich School of Philosophy, the excellence Center for Responsible AI Technologies aims to explore integrating philosophical, ethical, and social scientific perspectives into AI research. The primary objective is to contribute to the socially responsible development of AI technologies and to act as an innovation accelerator on three levels:

First, a focus is placed on research and methodological development. This includes concepts for practically integrating AI applications such as medicine, the future of work, mobility, and climate. It encompasses a holistic view of inter- and transdisciplinary research that deals with AI's philosophical-ethical and societal aspects. Second, the center focuses on educating and training the next generation of professionals and decision-makers. By delving into AI's ethical and societal challenges, it aims to enhance future experts' awareness and technical skills. Third, importance is placed on stimulating societal discourse and outreach. This involves including diverse societal perspectives and groups to further promote the acceptance and understanding of AI within society.(Hochschule für Philosophie München / Philosophische Fakultät 2022)

**Personal Contribution:** The participation in the Center for Responsible AI, which is related to this thesis, mainly consisted of knowledge transfer in the form of talks and workshops.

### AI Production Network

**Summary:** The AI Production Network was initiated to future-proof the economy through advanced research in the field of AI for the production sector. The Production Network is dedicated to researching AI solutions that are tested on production facilities on an industrial scale. Within the network, organizational units have been established that focus on education, further training, and the development of a shared research infrastructure. Through collaborations with academic partners and small and medium-sized enterprises as well as large industrial partners, the AI Production Network sets research priorities in various areas. These include resilient material technologies and value creation networks, generative design methods and material development, as well as adaptive manufacturing processes and closed-loop production. Furthermore, intensive research is conducted on digital twins for products, materials, processes, and production networks. Another area of focus is human-centered production technologies and self-organizing process route planning, aiming to optimize human-machine interaction and the efficiency of production processes.

**Personal Contribution:** Within the AI production network, parts of concepts from this dissertation could be integrated into project deliverables. Concrete participation included the development of new ideas and the supplementation of existing concepts. Based on expertise in

AL, MLOps, and XAI, results contributed to several projects' evolution. Further engagement took place in the context of workshops, especially intending to transfer the know-how in the previously mentioned areas to future scientists.

### 1.5.2 Supervised Theses

During the elaboration of this research, several Bachelor's and Master's theses were supervised, for which the author of this dissertation developed the topics and conceptual decisions. These studies contributed to broadening the scope and depth of this dissertation. A selection of these works served as independent research projects and as components that complement the overarching narrative of this dissertation.

The following section provides a brief overview of the relevant theses and places them in the context of this dissertation.

**Reduction of Annotation Effort - A Comparison of Active Learning Frameworks**
(Albrecht 2020)

**Summary:** In this thesis, with its original title *„Reduktion des Annotationsaufwands - Ein Vergleich von Active Learning Frameworks"*, various AL frameworks are introduced and systematically compared. The repositories of open-source projects of different AL tools were evaluated to assess the licensing, support, and usability. To measure its performance, AL simulations were conducted, and benchmark tests were implemented. Furthermore, the frameworks were evaluated concerning their flexibility and extensibility, and the results were analyzed using a utility value analysis.
**Related sections in thesis:** Part III, Chapter 5

**Bias Detection using Interpretable Machine Learning Methods**
(Figel 2021)

**Summary:** This thesis addresses the growing awareness and accompanying concerns regarding the potential bias of ML models. It outlines the field of IML, advocates for using these methods to make models more transparent, and investigates how they can contribute to uncovering bias. To this end, experiments are initiated in which a deliberate synthetic bias is introduced into the analyzed data. Subsequently, using IML techniques, it is evaluated how effectively they are suited for bias detection.
**Related sections in thesis:** Part IV, Chapter 7

**Evaluating Explainability in the Context of Active Learning**
(Elia 2021)

**Summary:** This Master's thesis combines the methodologies AL and IML. The investigation focuses on the dynamics that underlie the model and their consequential effects on the output of the model interpretations. Specifically, domain-specific explanations for a skin image analysis classification model are generated in an AL simulation and evaluated

according to the requirements of a reasonable explanation accuracy, degree of importance, novelty and representativeness, stability, consistency, certainty, and fidelity.
**Related sections in thesis:** Part IV, Chapter 7

### Model Explanations as Quality Gate in Machine Learning Pipelines

(Bergmair 2022)

**Summary:** This Master's thesis lies at the intersection of IML methods and MLOps and presents the conception of an automated quality check of an ML model based on the results of an IML method. Besides computing performance metrics to evaluate the predictive performance of a trained ML model on a test dataset, an additional step aims to measure the quality of the prediction in terms of interpretability. By ensuring that the model makes valid statements before it is released, this additional step is intended to support an automated release process. Specifically, an ML pipeline was implemented for an image classification task, where a separate stage uses the trained model and validates the generated predictions by applying interpretation methods and calculating a quality measure.
**Related sections in thesis:** Part III, Chapter 5 and 6

### Interpretable Machine Learning with ECG Data

(Trautwein 2022)

**Summary:** The application of different interpretation methods to a 12-lead ECG classifier is implemented and investigated in this thesis. The goal is to adapt the standard implementations to the domain-specific requirements of the related use case. For this purpose, on the one hand, different interpretation techniques and, on the other hand, their output with different representation forms were investigated. In this context, an initial concept for the visualization of the model explanations was developed, and the first visual details were investigated.
**Related sections in thesis:** Part IV, Chapter 8

### Evaluating Explainable AI Algorithms Using Scenario-Based Focus Groups

(Lang 2023)

**Summary:** This Master's thesis delves into the multifaceted perspectives of stakeholders involved in the development of AI systems. Employing focus group discussions, a well-established methodology in social science, were used to develop a scenario related to the practical application of an AI-enhanced ECG classifier This scenario was then critically discussed within two specialized expert groups. The feedback and insights derived from these discussions were meticulously analyzed to comprehend the facets related to the trustworthiness of the AI system.
**Related sections in thesis:** Part IV, Chapter 8

**MLOps in Academia: From Concept to Implementation**
(Weigell 2024)

**Summary:** This Master's thesis investigates the integration of MLOps into academic ML research, a practice that is widely established in the industry but still scarcely adopted in the academic environment. The objective is to develop the Academic MLOps Framework, which is created by adapting MLOps principles to the academic context, based on an interview study and a multivocal literature review. This framework comprises the Academic MLOps Platform, realized through a reference implementation using a Kubernetes cluster and open-source tools as well as a tailored workflow. The combination of these components aims to facilitate the application of MLOps in academic research. This work contributes to bridging the methodological gap between the industry and academic ML research by successfully integrating MLOps principles into the academic research process.
**Related sections in thesis:** Part II, Chapter 4 and Part II, Chapter 6

### 1.5.3 Publications

A central part of the scientific work within the context of this dissertation is the publications directly related to the research findings and insights of this thesis. The following Section introduces publications that underwent a peer-review process - a procedure ensuring that the presented concepts, implementations, results, and conclusions withstand critical examination by experts in the respective field and meet high academic standards.

Passages and findings from the author that were previously published and are utilized in this thesis are not additionally cited.

**Reference:**
(Stieler *et al.* 2021)

**Towards Domain-Specific Explainable AI: Model Interpretation of a Skin Image Classifier using a Human Approach**

**Authors:** Fabian Stieler, Fabian Rabe and Bernhard Bauer

**Conference:** IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 19-25, 2021, Nashville, TN, USA

**Abstract:** ML models have started to outperform medical experts in some classification tasks. Meanwhile, the question of how these classifiers produce certain results is attracting increasing research attention. Current interpretation methods provide a good starting point in investigating such questions, but they still massively lack the relation to the problem domain. In this work, we present how explanations of an AI system for skin image analysis can be made more domain-specific. We apply the synthesis of Local Interpretable Model-agnostic Explanations with the ABCD-rule, a diagnostic approach of dermatologists, and present the results using a Deep Neural Network based skin image classifier.

**Personal Contribution:** I designed and implemented the methodology, designed the experiments, formulated the hypotheses and analysed the results. I researched and wrote the main parts of the paper. I presented the paper at the CVPR2021 workshop.

**Text passages are included in this thesis:** Part IV, Chapter 7

### Git Workflow for Active Learning - A Development Methodology Proposal for Data-Centric AI Projects

**Reference:**
(Stieler *et al.* 2023a)

**Authors:** Fabian Stieler and Bernhard Bauer

**Conference:** 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), April 24-25, 2023, Prague, Czech Republic

**Abstract:** As soon as AI projects grow from small feasibility studies to mature projects, developers and data scientists face new challenges, such as collaboration with other developers, versioning data, or traceability of model metrics and other resulting artifacts. This paper suggests a data-centric AI project with an AL loop from a developer perspective and presents „Git Workflow for AL “: A methodology proposal to guide teams on how to structure a project and solve implementation challenges. We introduce principles for data, code, as well as automation, and present a new branching workflow. The evaluation shows that the proposed method is an enabler for fulfilling established best practices.

**Personal Contribution:** I researched and wrote the main parts of the paper. This concerns both the initially submitted version and the revision of the approved version. Furthermore, I presented the paper at ENASE2023.

**Text passages are included in this thesis:** Part II, Chapter 3 and 4

### Enhancing Collaboration and Agility in Data-Centric AI Projects

**Reference:**
(Stieler *et al.* 2024)

**Authors:** Fabian Stieler and Bernhard Bauer

**Book:** Evaluation of Novel Approaches to Software Engineering, 18th International Conference ENASE, April 24-25, 2023, Revised Selected Papers, in Springer Nature: Communications in Computer and Information Science

**Abstract:** Usually, mature Artificial Intelligence (AI) projects are developed by a team of various members, such as data engineers, data scientists, software engineers and machine learning (ML) engineers. They often pursue highly heterogeneous approaches, leading to new challenges in collaboration, particularly regarding software quality, data versioning and the traceability of model metrics and other resulting artifacts. These challenges are further intensified when AI projects rely on dynamic datasets, introducing an entirely new dimension that teams must deal with. Adopting principles from the machine learning operations (MLOps) paradigm becomes essential in this context. To go beyond existing process models and develop actionable guidelines, our work introduces a Git workflow for AI projects. We present basic instructions for data and code while outlining a minimal infrastructure setup. Building upon abstract concepts, we delve into concrete, actionable steps by examining the proposed branching workflow. Through a case study, we apply the development methodology to two use cases and demonstrate that the principles and approaches positively impact project outcomes.

**Personal Contribution:** I researched and wrote the main parts of the paper. This concerns both the initially submitted version and the revision of the approved version.

**Text passages are included in this thesis:** Part II, Chapter 3 and 4

**Reference:**
(Stieler *et al.* 2023b)

### LIFEDATA - A Framework for Traceable Active Learning Projects

**Authors:** Fabian Stieler, Miriam Elia, Benjamin Weigell, Bernhard Bauer, Peter Kienle, Anton Roth, Gregor Müllegger, Marius Nann and Sarah Dopfer

**Conference:** 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW), 4-5 September 2023, Hannover, Germany

**Abstract:** AL has become a popular method for iteratively improving data-intensive AI models. However, it often presents a significant challenge when dealing with large volumes of volatile data in projects, as with an AL loop. This paper introduces LIFEDATA, a Python- based framework designed to assist developers in implementing AL projects focusing on traceability. It supports seamless tracking of all artifacts, from data selection and labeling to model interpretation, thus promoting transparency throughout the entire model learning process and enhancing error debugging efficiency while ensuring experiment reproducibility. To showcase its applicability, we present two life science use cases. Moreover, the paper proposes an algorithm that combines query strategies to demonstrate LIFEDATA's ability to reduce data labeling effort.

**Personal Contribution:** I researched and wrote the main parts of the paper. This concerns both the initially submitted version and the revision of the approved version.

**Text passages are included in this thesis:** Part III, Chapter 5 and 6

**Reference:**
(Ziethmann *et al.* 2024)

### Towards a Framework for Interdisciplinary Studies in Explainable Artificial Intelligence

**Authors:** Paula Ziehtmann, Fabian Stieler, Raphael Pfrommer, Kerstin Schlögl-Flierl, Bernhard Bauer

**Conference:** 26th International Conference on Human-Computer Interaction (HCII2024), 29. June - 04. July 2024, Washington DC, USA

**Abstract:** In this interdisciplinary paper, we present the SocioTechXAI

Integration Framework (STXIF), a novel approach that aims to seamlessly integrate technical advances with social science methodologies for a nuanced understanding of Explainable Artificial Intelligence (XAI) tailored to specific use cases. We begin with an overview of related work exploring XAI in both the social sciences and computer science. The focus is on the presentation of the STXIF, which includes the XAI-Compass and socioscientific analysis of stakeholder perspectives. The perspectives of these stakeholders, classified by the XAI-Compass as Model Breakers, Model Builders, and Model Consumers, are investigated using qualitative content analysis. By examining the EU AI Act as a Model Breaker and conducting scenario-based focus group discussions with Model Consumers (medical professionals) and Model Builders (developers) in a real medical diagnostic use case, we demonstrate the specific insights gained through the STXIF application and its adaptability in real scenarios. The discussion section addresses the complex relationships between different XAI goals and their implications for a flexible and adaptive development approach. The practical implications extend to the concrete development and implementation of XAI in real-world applications, in line with the thematic focus of the Human Computer Interaction International Conference, which emphasizes human-centered design and usability in interactive systems. Emphasizing nuanced interactions with XAI and its practical applications establishes a foundational framework for future interdisciplinary research and application in the evolving landscape of human-computer interaction (HCI).

**Personal Contribution:** I was involved in structuring the paper. Furthermore, i designed and realised the scenario-based focus group discussions and designed the XAI-Compass. I wrote parts of the introduction and the related work section, as well as the section of the XAI-Compass.

**Text passages are included in this thesis:** Part IV, Chapter 8

**Further Publications** In addition to the core research findings presented in this dissertation, other publications have been contributed to during the course of this PhD study, which are listed below.

- **Stieler, F.**, Rabe, F., & Bauer, B. (2020). Federated medical data - how much can deep learning models benefit? *AMIA 2020 Virtual Clinical Informatics Converence*, May 19-21, 2020, Seattle, WA, USA

**Reference:**
(Stieler *et al.* 2020)

- Elia, M., Peter, T., **Stieler, F.**, Bauer, B., Nagl, S., Ebigbo, A., & Grünherz, V. (2023). Precision medicine for achalasia diagnosis: a multi-modal and interdisciplinary approach for training data generation, *IEEE - ISBI 2023: International Symposium on Biomedical Imaging*, April 18-21, 2023, Cartagena de Indias, Colombia

**Reference:**
(Elia *et al.* 2023)

- Nagl, S., Grünherz, V., Elia, M., Peter, T., **Stieler, F.**, Bauer, B., Messmann, H. & Ebigbo, A. (2023). Automatische dreidimensionale Rekonstruktion des Ösophagus zur Vorhersage des Therapieerfolges bei Patienten mit Achalasie, *Zeitschrift für Gastroenterologie*, 61(08), KV259

**Reference:**
(Nagl *et al.* 2023)

- Elia, M., **Stieler, F.**, Ripke, F., Nann, M., Dopfer, S., & Bauer, B. (2024). Towards Certifiable AI in Medicine: Illustrated for Multi-label ECG Classification Performance Metrics, *Proceedings of the IEEE International Conference on Evolving and Adaptive Intelligent Systems 2024*, Madrid, Spain

**Reference:**
(Elia *et al.* 2024)

- Nagl, S., Grünherz, V., Elia, M., **Stieler, F.**, Peter, T., Bauer, B., Muzalyova, A., Messmann, H., & Ebigbo, A. (2024). Automatic Three-Dimensional Reconstruction of the Esophagus in Achalasia Patients undergoing POEM: a Comprehensive Assessment of Treatment Outcomes and pathophysiological Changes. *Endoscopy*, 56(S 02):MP085

**Reference:**
(Nagl *et al.* 2024)

- Grünherz, V., Ebigbo, A., Elia, M., Brunner, A., Krafft, T., Pöller, L., Schneider, P., **Stieler, F.**, Bauer, B., Muzalyova, A., Messmann, H., Nagl, S. (2024). Automatic three-dimensional reconstruction of the oesophagus in achalasia patients undergoing POEM: an innovative approach for evaluating treatment outcomes. *BMJ Open Gastroenterology*, 11(1):e00139

**Reference:**
(Grünherz *et al.* 2024)

**Publications under Review**

- Weigell, B., **Stieler, F.**, & Bauer, B.; All You Need is an AI Platform: A Proposal for a Complete Reference Architecture
- Kranz, S., Ziehtmann, P., Hartmann, D., **Stieler, F.**, Bauer, B., & Schlögl-Flierl, K.; Integrating Citizen Perspectives in the Development of AI-Based Shared Decision Making for Basal Cell Carcinoma Diagnosis and Treatment: Insights from the OCTOLAB Project in Augsburg, Germany
- Lorenz, A., Bauer, B., Krafft, T. D. & **Stieler, F.**; Verantwortungsvolle Künstliche Intelligenz - Ein praktischer Leitfaden für die Umsetzung des EU AI Acts

# Foundations 2

**2.1 Machine Learning . . . . 24**
2.1.1 Learning Types . . . . . . 24
2.1.2 Artificial Neural Network 26
2.1.3 Active Learning . . . . . . 31
**2.2 Trustworthy Artificial Intelligence . . . . . . . . 38**
2.2.1 Ethical Principles . . . . . 39
2.2.2 Key Requirements . . . . . 40
**2.3 Explainable Artificial Intelligence . . . . . . . . 43**
2.3.1 Interpretable Machine Learning . . . . . . . . . . 43
2.3.2 Methods for Local Post-hoc Explanations . . . . . . . . 46
**2.4 Software Engineering for AI Systems . . . . . . . . . 49**
2.4.1 Process Models for Data-intensive Systems . . . . . 49
2.4.2 Operation Frameworks . . 51

In this chapter, fundamental concepts from various disciplines are introduced, which are significant for the subsequent chapters of this dissertation. Given the thematic diversity, this chapter does not claim to provide an exhaustive summary of all fundamentals. Instead, the presentation specifically focuses on those aspects essential for a thorough understanding of the following chapters.

Figure 2.1 provides a schematic overview of the connections between different research areas and their subfields, as well as potential intersections. Furthermore, this visualization clarifies which fundamental concepts become relevant in various parts of the dissertation.

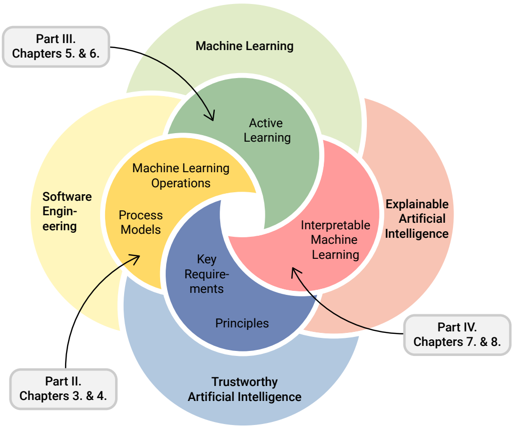


**Figure 2.1:** Research fields and their relevance to the thesis' parts

The thematic framework of this work is initially established in Section 2.1 with a brief introduction to the field of ML and deepens the technical details of the AL methodology. Building on this, Section 2.2 primarily explains the principles and key requirements associated with the development of trustworthy AI. In order to do justice to the focus on transparency and explainability in processes and systems, Section 2.3 provides a brief overview of the field of XAI and IML, while, finally in Section 2.4 the basics of SE for AI systems relevant to this thesis will be presented.

## 2.1 Machine Learning

The field of ML, which is significantly shaped by the disciplines of AI, computer science, and particularly statistics, primarily focuses on developing algorithms that enable computers to learn from data and make predictions or decisions based on that data. Murphy 2012 defines ML as

> *„ [...] a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (such as planning how to collect more data!).“*

ML deals with designing and implementing systems that can learn from experiences (the data) to solve a problem rather than relying on programming with explicit individual instructions.

James *et al.* 2013 describes a system generically through the simplified expression

$$y = f(x) + \epsilon \tag{2.1}$$

where $y$ represents the quantitative response of the system, the dependent variable, $x$ represents the input data (independent variables), and $\epsilon$ denotes the error term. This error term captures the random deviations from the fixed but unknown function $f$, due to model uncertainties and unaccounted-for variables, and tends to average to zero. ML methods are used to generate a model that approximates $f$, thereby producing predictions

$$\hat{y} = \hat{f}(x) \tag{2.2}$$

The accuracy of these predictions $\hat{y}$ for $y$ is influenced by both reducible and irreducible errors. The irreducible error remains due to factors such as unobserved variables and inherent variability, which cannot be predicted or controlled by the model. The goal of ML is to minimize the reducible error by applying appropriate learning techniques to estimate $f$.(James *et al.* 2013)

### 2.1.1 Learning Types

A typical differentiation of ML methods arises from the type of feedback available during the algorithm's learning phase. This feedback influences the strategy with which the function $f$ is approximated and optimized, leading to a classification into three basic forms of learning types:

In **1) Reinforcement learning**, learning does not occur directly from datasets; instead, various strategies are tried in a dynamic environment, where the trained model learns from the outcomes of the actions taken.

**2) Unsupervised learning** is applied when there are no corresponding responses $y_i$ for the observations $i = 1..n$ in the input vector $x_i$ within the training data. In this case, the goal is to discover structures or patterns within the data itself, which is useful for tasks such as clustering or dimensionality reduction.(Murphy 2012)

While reinforcement learning is not the subject of this thesis and unsupervised learning plays a subordinate role, the third and most common form of learning is crucial for AL.

In **3) Supervised learning**, the fundamental goal is to derive a function $f$ that maps an input vector $x$ to an output $y$, utilizing a labeled dataset $D_L = (x_i, y_i)_{i=1}^N$, where $N$ denotes the number of training examples. Each element of the dataset comprises an input feature vector $x_i$, which can vary in complexity from simple numerical values, such as height and weight, to structured objects like images, sentences, graphs, or signals. The corresponding output $y_i$ can assume various forms, including ordinal or categorical labels, ranging from simple binary classifications to more complex multi-class scenarios.(Murphy 2012)

In essence, supervised learning algorithms strive to learn this mapping function from the training data provided, enabling the model (within an AI system) to predict outputs for new, unseen inputs. The approximation function, as outlined in (2.2), is employed to predict inputs $x$ that were not included in the training set $D_L$.(Murphy 2012)

Supervised learning is primarily divided into two types of tasks, depending on the nature of the output variable $y$. James *et al.* 2013 describe these as follows:

- **Regression:** Here, the output variable $y$ is continuous, and the method models the relationship between the input data and a continuous outcome. Typical applications include estimating real estate prices or predicting temperatures in weather forecasting.
- **Classification:** Conversely, classification addresses tasks where the output variable $y$ is categorical. The model must determine the appropriate category or class for the data based on the given input data $x$. These categories can be, for example, medical diagnoses such as explored in the use cases detailed in Chapter 6.

The ability to achieve good results on new, unseen data is referred to as generalization. A supervised learning algorithm assesses its performance using a loss function that measures the errors between the probabilistic output $\hat{y}$ and the actual response $y$. The central challenge in supervised learning is to optimize the models' predictive performance by minimizing the discrepancy between $\hat{y}$ and $y$. This is achieved by minimizing the loss function $\mathscr{L}(y, \hat{y})$, which quantifies the errors for each training example.(James *et al.* 2013)

We continue and focus on the classification task type as it is relevant to this thesis. When the output $y$ is categorical, a specific form of the loss function, known as Categorical Cross Entropy, is particularly pertinent. It is suitable when multiple classes exist and $f(x)$ estimates a probability for each class:

$$\mathscr{L}(y, \hat{y}) = -\sum_{n=1}^{N} \sum_{c=1}^{C} y_{nc} \log(\hat{y}_{nc}), \tag{2.3}$$

where $\hat{y}_{nc}$, is an indicator that specifies whether observation $n$ belongs to class $c$, and $C$ is the number of classes.(James *et al.* 2013)

### 2.1.2 Artificial Neural Network

Among the various methods for implementing $f()$ within supervised learning, such as regression models, support vector machines, and decision trees, which differ in their analytical as well as computational properties, artificial neural networks have gained prominence in recent years due to their flexible architecture and ability to handle non-linear relationships. Given their relevance to this thesis, the following will discuss artificial neural networks in more detail.

#### Basic Structure and Concepts

Initially inspired by biological systems, artificial neural networks have proven particularly effective in recent years due to their flexible architecture and ability to process non-linear relationships. Biological neural networks consist of interconnected cells that process and communicate signals with each other.(Bishop 2006; Charniak 2018)

An artificial neural network adapts this concept and essentially consists of connected nodes and artificial neurons. These are organized into an input layer, one or more hidden layers, and an output layer, as illustrated in Figure 2.2. Each neuron in a layer receives inputs, applies a weighted sum followed by a non-linear activation function, and forwards the result to the next layer. To elucidate the functioning, we shall commence with the basic form of an artificial neural network:

**The Perceptron.** The simplest component of an artificial neural network is the perceptron, a single-layer neural network. It consists of a vector of weights $w = [w_1..w_m]$, one for each input feature $x_i$, and a distinguished weight $b$, called bias. Let $\Phi = \{w \cup b\}$ be the parameters of a perceptron with $\phi_i$ as $i$th parameter, a perceptron receives inputs and calculates the following function:

$$f_\Phi(x) = \begin{cases} 1 & \text{if } b + \sum_{i=1}^{l} x_i w_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Rephrased, the perceptron's operation involves taking each input, multiplying it by its corresponding weight, and adding a bias term to the sum. If the resulting value exceeds zero, the output is set to 1; otherwise, it is set to 0. Using the standard notation, where the dot product of two vectors of length $l$ is defined as

$$x \cdot y = \sum_{i=1}^{l} x_i y_i \tag{2.5}$$

lead to an expression for the perceptron's calculation as follows:

$$f_\Phi(x) = \begin{cases} 1 & \text{if } b + w \cdot x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

The perceptron can classify linearly separable data. However, its capacity to solve more complex problems is limited by its linear decision boundary.(Charniak 2018)

**The Neuron.** Building upon the perceptron, a single neuron in a more sophisticated network can be described by a similar weighted sum and activation function. This neuron can be seen as a mathematical model that processes input signals through a series of weights and biases. The neuron's output is a function of the weighted sum of its inputs, passed through a non-linear activation function $\sigma$:

$$\hat{y} = \sigma\left(\sum_j w_j x_j + b\right) \tag{2.7}$$

The adjustment of the weights $w$ and the bias $b$ in an artificial neuron occurs through a learning process in which the parameters are altered to minimize the error function $\mathscr{L}(y, \hat{y})$. This is done using an optimization method, like gradient descent, which incrementally adjusts the parameters in the direction of the steepest descent of the error gradient. The gradient of the error is calculated as follows:

$$\nabla\mathscr{L} = \frac{\partial\mathscr{L}}{\partial w}, \quad \nabla\mathscr{L}_b = \frac{\partial\mathscr{L}}{\partial b} \tag{2.8}$$

Then, the weights and biases are updated by:

$$w = w - \eta\nabla\mathscr{L}, \quad b = b - \eta\nabla\mathscr{L}_b \tag{2.9}$$

where $\eta$ is the learning rate, a real number that determines the extent to which a parameter is adjusted during a given update. The gradient indicates how significantly a small change in each weight and bias affects the error. This process is iteratively repeated for many training examples to systematically improve the model.(Ruder 2017)

An advanced form of gradient descent is Adaptive Moment Estimation (Adam), proposed by Kingma *et al.* 2015, which utilizes adaptive learning rates for different parameters. Adam combines the advantages of two other extensions of gradient descent: a parameter-wise adaptation of the learning rate and an adaptive adjustment of the learning rates based on the average first and second momentum of the gradient. The update rules for Adam are given by:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla\mathscr{L}, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla\mathscr{L})^2 \tag{2.10}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.11}$$

$$w = w - \eta\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.12}$$

Here, $m_t$ and $v_t$ are estimates of the first and second momentum of the gradient, $\beta_1$ and $\beta_2$ are decay factors for these momentums, and $\epsilon$ is a minimal value to prevent division by zero.(Kingma *et al.* 2015)

Building on the concept of Adam, Nesterov-accelerated Adaptive Moment Estimation (NAdam), proposed by Dozat 2015, incorporates Nesterov momentum into the Adam optimizer framework. NAdam enhances the traditional momentum method by calculating a look-ahead update for the parameters, which anticipates future gradients based on current momentum. This is particularly effective for reducing the oscillations in updates and speeding up convergence. While we retain the first update step of Adam (2.10), the adjustment of the first and second momentum is given by:

$$\hat{m}_t = \frac{\beta_1 m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.13}$$

as well as the parameter update incorporating Nesterov momentum:

$$w = w - \eta \frac{\hat{m}_t + \frac{(1-\beta_1)\nabla\mathcal{L}}{1-\beta_1^t}}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.14}$$

whereby the parameter update steps are performed based on the gradient, and a refined estimate of future gradients is taken into account.(Dozat 2015)

The activation function $\sigma$ used in the neuron plays a crucial role, as it introduces non-linearity, enabling the network to model complex functions and learn non-linear decision boundaries. Prominent activation functions are the Sigmoid and Rectified Linear Unit (ReLU), The Sigmoid function, defined by

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

is particularly effective for binary classifications due to its ability to map values into a $(0, 1)$ range. On the other hand, the ReLU function, represented by

$$\sigma(x) = \max(0, x) \tag{2.16}$$

is frequently used in deeper networks for its computational efficiency and effectiveness in promoting sparse activations, which can enhance the learning process in complex network architectures.(Goodfellow *et al.* 2016)

### Multilayer Neural Networks

Multilayer neural networks, the fundamental concept of DL, consist of multiple layers of neurons. Figure 2.2 illustrates such a structure, where the first layer is referred to as the input layer and the last layer as the output layer. All intervening layers are called hidden layers. These structures enable the networks to capture complex patterns in the data by

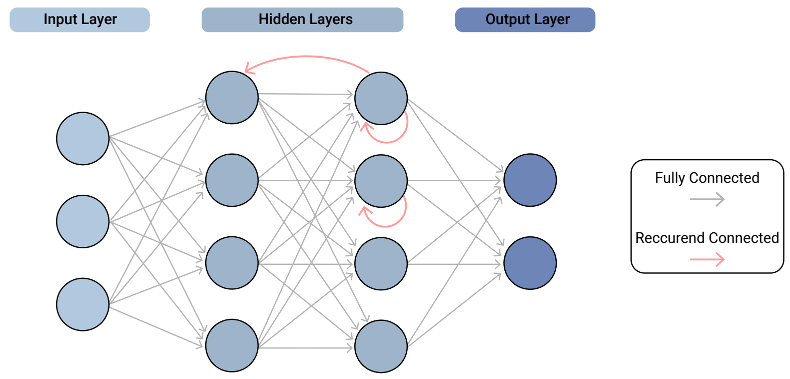


**Figure 2.2:** Schematic representation of a multi-layer neural network, consisting of an input layer, two hidden layers, and an output layer. The neurons within these layers are depicted as circles. They are interconnected by gray lines, illustrating a fully connected structure where each neuron in one layer is connected to all neurons in the subsequent layer. The red arrows highlight recurrent connections between some neurons in the hidden layers, representing a special case where the neurons or layers are able to retain information by maintaining a state.

using more than one hidden layer, allowing each layer to incrementally extract higher-value features from the inputs and thus represent the information. The number of layers and neurons and the number and characteristics of the connections between them can vary, which in this context describes the model architecture.

The simplest form are feedforward networks, suitable for illustrating the basic concept of multilayer neural networks. The general mathematical representation for a feedforward network involves a series of transformations:

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}, \quad a^{(l+1)} = \sigma(z^{(l+1)}) \tag{2.17}$$

where $z^{(l+1)}$ is the input to layer $l+1$, $W^{(l)}$ represents the weight matrix for layer $l$, $b^{(l)}$ is the bias vector, $a^{(l)}$ is the activation from the previous layer (with $a^{(0)} = x$ for input layer), and $\sigma$, again, denotes the activation function applied element-wise.(Goodfellow *et al.* 2016)

In contrast to feedforward networks, where information moves in only one direction – from input nodes through hidden layers to output nodes – Recurrent Neural Networks (RNN) have loops allowing information to be persisted. An RNN processes sequences by maintaining a state (memory) that captures information about what has been calculated so far. The basic formulation of an RNN can be expressed as follows:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \quad y_t = W_{hy}h_t + b_y \tag{2.18}$$

where $x_t$ is the input at time $t$, $h_t$ is the hidden state at time $t$, $W_{xh}$, $W_{hh}$, and $W_{hy}$ are weights, and $b_h$ and $b_y$ are biases.(Lipton *et al.* 2015)

This recurrent structure endows RNNs with the capacity to cultivate a profound comprehension of sequences and their contextual frameworks, distinguishing them from feedforward networks. Nevertheless, despite their inherent capabilities, traditional RNNs frequently encounter difficulties in sustaining long-term dependencies, a phenomenon exacerbated by the vanishing gradient problem. In response to these challenges, Long Short-Term Memory (LSTMs), conceptualized by Hochreiter *et al.* 1997, introduce an enhanced architectural paradigm. Applied to an RNN, LSTMs incorporate a sophisticated gating mechanism – encompassing input, output, and forget gates – that meticulously regulates the flow of information, thereby empowering the model to dynamically retain or

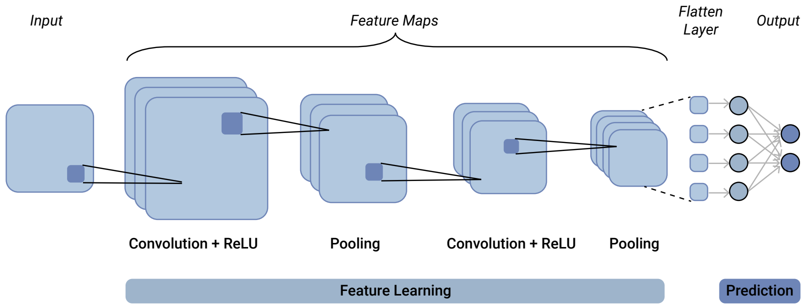


**Figure 2.3:** Visual depiction of a convolutional neural network. This multi-layered model begins with an input layer, followed by alternating convolutional layers with ReLU activation and pooling layers for spatial reduction. The layers extract features, represented by varied depths in the „Feature Maps“. A flattening layer converts these maps into a one-dimensional vector, which feeds into a fully connected network, culminating in the output.

expunge information across extended sequences, thus ensuring stability and augmenting performance.

While RNNs excel in sequence prediction and handling time-series data, Convolutional Neural Networks (CNNs), as illustrated in Figure 2.3, are uniquely adept at processing spatial data. This specialization makes CNNs indispensable for applications involving grid-like data structures, such as images and videos, where preserving and interpreting spatial relationships are crucial.

Convolution, a fundamental mathematical operation in CNNs, extracts features from input data by applying filters or kernels. Following the notation from Goodfellow *et al.* 2016, the convolution for a given input function $x(t)$ and a kernel $w(t)$ is:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\,da \tag{2.19}$$

In practical terms, especially in digital applications like image processing, convolution involves sliding the kernel over the input, calculating the dot product at each position. This process efficiently captures local dependencies and patterns in the input data.

For a two-dimensional input, such as an image $I$, and a two-dimensional kernel $K$, the discrete convolution is given by:

$$S(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m,n)K(i-m, j-n) \tag{2.20}$$

This operation is applied repeatedly across the image, allowing the network to build a complex hierarchy of features, from simple edges to more abstract shapes and structures.(Goodfellow *et al.* 2016)

Each convolution layer is typically followed by a ReLU activation layer, which introduces non-linearity, enhancing the network's ability to learn complex patterns. As shown in Figure 2.3, this is followed by pooling layers, which reduce the spatial dimensions of the feature maps, thereby decreasing computation and memory usage while still preserving essential features. These layers collectively form the feature learning phase of a CNN, which culminates in a flatten layer that prepares the feature-rich data for the final prediction phase, effectively translating the intricate spatial hierarchies into outputs that the network can use for classification or other tasks.(O'Shea *et al.* 2015)

### 2.1.3 Active Learning

As described in Section 2.1.1, the training of ML models following the principle of supervised learning requires large quantities of labeled datasets ($D_L$). Often, data are available in an unlabeled form ($D_U$), characterized by the absence of the label $y_i$ for a corresponding feature vector of sample $x_i$. Labeling this data can present various challenges, particularly involving human annotation. For instance, rare specialized expertise may be required, or the time and involvement of experts may lead to high costs.(Tharwat *et al.* 2023)

AL represents a specialized subset of ML where the model actively intervenes in the process of data selection to enhance its performance. Unlike traditional approaches in supervised learning that 'passively' receive a fully labeled dataset, AL strategically selects unlabeled data points to be labeled. The most cited survey paper by Settles 2009 provides a definition of AL according to which

> *„Active learning systems attempt to overcome the labeling bottleneck by asking queries in the form of unlabeled instances to be labeled by an oracle (e.g., a human annotator). In this way, the active learner aims to achieve high accuracy using as few labeled instances as possible, thereby minimizing the cost of obtaining labeled data."*

AL addresses the challenges of data labeling by selecting a small yet targeted subset of unlabeled data for annotation, thus making the training of ML models more effective. In addition to the primary goals of achieving high accuracy and reducing labeling efforts, AL provides the framework for implementing targeted search strategies for specific types of data or information within a larger dataset (Tharwat *et al.* 2023). Ghai *et al.* 2021 summarize that beyond the original definition of AL focusing on instance labels, the method has been used to query feedback on features, for example, to determine whether the presence of a feature is an indicator of the target concept (Druck *et al.* 2009; Raghavan *et al.* 2006; Settles 2011), an active class selection (Lomasky *et al.* 2007), or even active feature acquisition (Zheng *et al.* 2002).

#### Scenarios

AL represents an interactive training paradigm in which Settles 2009 delineates three distinct problem scenarios wherein the learner (the model) is permitted to pose queries. Figure 2.4 delineates the distinctions among these three principal scenarios, predicated on the assumption that the queries are unlabeled instances to be labeled by an oracle.

**Membership Query Synthesis.** Membership Query Synthesis, first investigated by Angluin 1988, allows the model to generate synthetic instances in the input space independently and subsequently request their labeling. This approach is particularly suitable for finite problem domains, as it does not require the processing of unlabeled data and enables the learner to generate query instances quickly. The advantage of this method lies in scenarios where data can be easily synthesized. Thus,

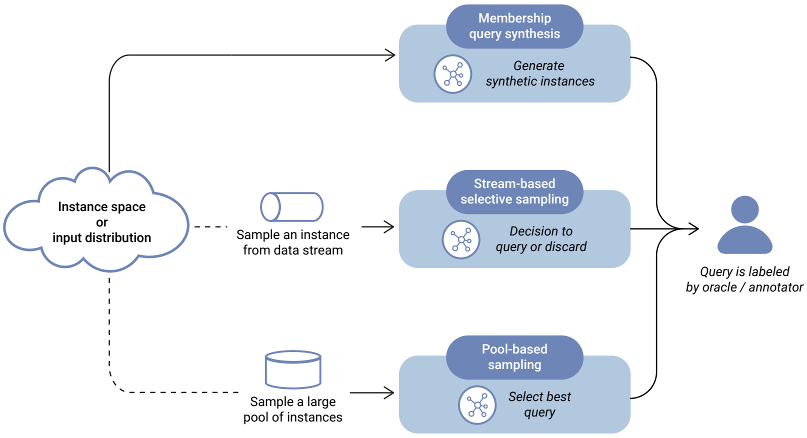

Membership query synthesis
Generate synthetic instances
Stream-based selective sampling
Decision to query or discard
Pool-based sampling
Select best query
Instance space or input distribution
Sample an instance from data stream
Sample a large pool of instances
Query is labeled by oracle / annotator

**Figure 2.4:** Three different Active learning scenarios based on Settles 2009

promising results have been shown in non-human applications, such as the „robotic scientist" described in King *et al.* 2004 and King *et al.* 2009, which conducts independent biological experiments, demonstrating the potential of this approach in technical and scientific domains.(Settles 2009)

Despite the effectiveness of this approach, there are limitations, particularly the generation of instances whose labeling poses a challenge in itself. For example, in the work of Lang *et al.* 1992, the synthetic generation of images for the classification of handwritten characters resulted in many images that contained no recognizable symbols and were thus difficult for human annotators to interpret. This aspect reveals the limited applicability of the Membership Query Synthesis approach, especially when the oracle is a human annotator faced with artificially generated instances that lack natural semantic meaning.(Settles 2009)

**Stream-Based Selective Sampling.** Stream-based selective sampling, introduced by Atlas *et al.* 1989 and Cohn *et al.* 1994, operates under the assumption that acquiring an unlabeled instance is cost-effective, thus allowing the instance to be sampled from the actual distribution. Subsequently, the learner evaluates each sample to decide whether to request a label. This process, often referred to as sequential or stream-based AL, involves assessing each individual instance from the data stream and deciding whether to label or discard it. The uniqueness of stream-based sampling lies in its reliance on the natural distribution of data, ensuring that even when the distribution is non-uniform or unknown, the generated queries remain relevant and grounded in real scenarios.(Settles 2009)

The relevance and utility of stream-based selective sampling have been confirmed through various applications. In several studies (e.g., Thompson *et al.* 1999; Moskovitch *et al.* 2007), the scenario of selective sampling has been considered as a type of pool-based scenario, which is described subsequently. Both scenarios share the commonality of deriving instances from a real data distribution, with the main difference being that in selective sampling, the data are sampled sequentially, while in pool-based sampling, a large set of data points are sampled (Tharwat *et al.* 2023).(Settles 2009)

**Pool-Based Sampling.** Pool-based sampling, motivated by the frequent presence of large volumes of unlabeled data in real-world problems, is the most prominent scenario in AL. Introduced by Lewis *et al.* 1994b, this scenario assumes the existence of a small set of labeled data $D_L$ and a large pool of unlabeled data $D_U$. By employing a query strategy (QS), the selection of data points to be labeled from the pool $D_U$ is assessed and presented to the oracle for labeling. After the oracle labels these data points, the newly labeled data are incorporated into $D_L$, the training dataset, and used for training the model.(Settles 2009)

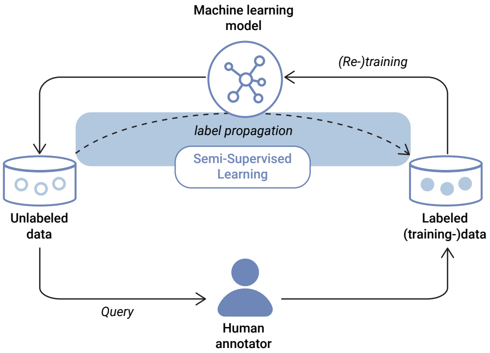


**Figure 2.5:** Pool-based Active learning cycle based on Settles 2009

As illustrated in Figure 2.5, the essence of this scenario lies in its cyclical process, where many or all instances in the pool are iteratively evaluated. This procedure continues until a predefined termination condition is met, such as exhausting a specific query budget or determining that no significant improvements can be made to the model's performance. The iterative nature of evaluating and expanding the training dataset underpins the dynamic learning process in pool-based sampling, making it particularly effective for refining model performance in data-centric environments.(Tharwat *et al.* 2023)

While in pool-based sampling AL scenarios, human annotators may be queried for labels, semi-supervised learning techniques (Yarowsky 1995) provide an alternative potential for effective data labeling. As depicted in Figure 2.5, the unlabeled data are used to further enhance the model trained from the labeled data $D_L$. Here, the learner identifies unlabeled data points from $D_U$ that are likely to be correctly predicted. These instances are then assigned a pseudo-label and added to $D_L$, where they are available in the next training iteration.(Zhu 2008)

### Query Strategies

In Section 2.1.1, the concept of supervised learning was introduced, wherein a loss function $\mathscr{L}$ is used to reduce the training error with the aim of approximating a function $f$ as accurately as possible. However, since the test error for unseen data cannot be directly computed during training, the process is often described as selecting a hypothesis with the lowest empirical risk from the possible hypothesis space.(Tharwat *et al.* 2023)

In the AL context, the core question is how the active learner can efficiently cover the hypothesis space using a limited querying budget. Unlike conventional supervised learning, where the sample of observations is

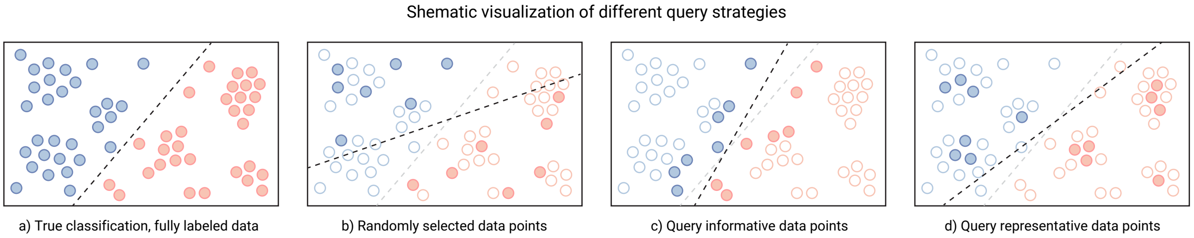

Shematic visualization of different query strategies
a) True classification, fully labeled data
b) Randomly selected data points
c) Query informative data points
d) Query representative data points

**Figure 2.6:** Schematic visualization of different QSs, based on Settles 2009 and Tharwat *et al.* 2023: a) A fictional fully labeled dataset with a decision boundary of a possible classifier. In b) the model is trained with a sample of labeled instances randomly drawn from the problem domain, which results in a less accurate decision boundary of the classifier. In c) the selection was made specifically for informative instances, and in d) for representative instances. In both cases, the decision boundary approaches its optimum but shows that the QSs lead to different classifiers.

randomly selected, AL uses strategic queries to minimize empirical risk by improving the representation of the data space and thus enabling a more accurate approximation of $f$. This is achieved by selecting informative and representative data points, as illustrated schematically in Figure 2.6.(Tharwat *et al.* 2023)

Each QS utilizes these functions to assess the potential utility of points in the unlabeled dataset $D_U$ and guide the selection process to ensure coverage of the data space according to the desired goal and minimize extrapolation errors.

Over the past few years, several surveys have been published that provide comprehensive overviews of different QSs (Settles 2009; Fu *et al.* 2013; Kumar *et al.* 2020; Tharwat *et al.* 2023). Kumar *et al.* 2020 and Tharwat *et al.* 2023 present possible taxonomies, simplified summarized in Figure 2.7. Herein, the QSs are categorized based on their utility function, which is used to evaluate the utility values of a data instance from $D_U$ to be queried.

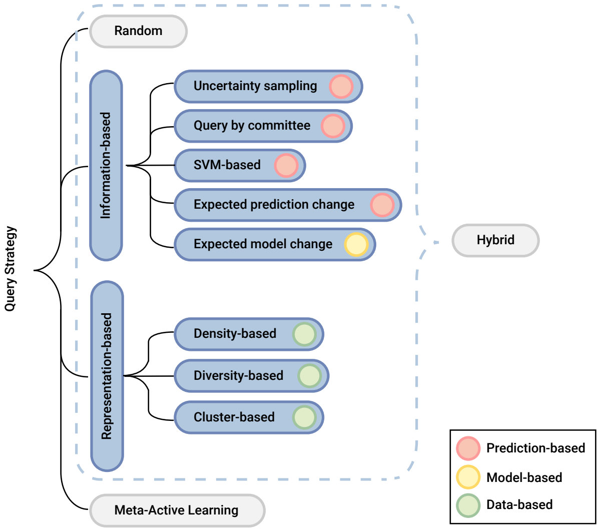


**Figure 2.7:** Taxonomy of QSs based on Kumar *et al.* 2020 and Tharwat *et al.* 2023.

A commonly used distinction in utility functions is the search for informative data points, as illustrated by their proximity to the decision boundary in Figure 2.6 c), and the search for representative data points, exemplified by the centers of potential clusters in Figure 2.6 d).(Tharwat *et al.* 2023)

Alternatively, QSs can be categorized based on the available information. Data-based strategies require the slightest knowledge, operating only with raw data and, in some cases, the labels of currently labeled data. In model-based strategies, in addition to the data, the model is required, but not its predictions. An example of this is the expected model change, a strategy where the data points queried are those that would have the most significant impact on model weights (Vezhnevets *et al.* 2012) or gradient length (Zhang *et al.* 2017). Prediction-based strategies require the most knowledge, as they need data, the model, and its predictions.(Tharwat *et al.* 2023)

Besides the main categories included in Figure 2.7, there are approaches often summarized as Meta-AL. These methods address the interdependence of the model's predictive performance and the QS by allowing the AL system to change the utility function during runtime, such as in the „AL by Learning" algorithm introduced by Hsu *et al.* 2015. In hybrid approaches, various utility functions are combined within a QS, aiming to utilize their respective advantages and enhance efficiency.

Following this, a selection of common QSs will be introduced, which will be relevant throughout this thesis.

**Information-based**

Information-based QSs aim to identify data points with the highest informational content to minimize the uncertainty of the model by specifically targeting those points that exhibit the greatest ambiguity. Typically, these points are located near the decision boundaries of the model, where predictions are most uncertain.

The primary motivation of these approaches is to enhance model accuracy by systematically requesting the labeling of instances from the oracle that most challenge and expand the existing function of the model. Consequently, the utility function used in these QSs is designed to specifically measure the uncertainty of each unlabeled data point to determine its potential informational value.

Settles 2009 describes uncertainty sampling (Lewis *et al.* 1994b) as possibly the simplest and most commonly used QS. This strategy selects instances for which the model's prediction probability is the lowest, making it suitable for probabilistic models where decision-making involves estimating the likelihood that a specific label is correct.

In a multi-class problem, a general variant of uncertainty sampling is applied, where the selection of most informative instances $x^*$ occurs at the point where the model has the least confidence in its most likely prediction:

$$x^* = \arg\max_x \left(1 - P_\theta(\hat{y}|x)\right), \tag{2.21}$$

where $\hat{y}$ is identified as the class label for which the model $\theta$ assigns the highest posterior probability given the input $x$.(Settles 2009)

Uncertainty sampling has proven particularly effective in applications involving sequential data, such as natural language processing and information extraction. It efficiently calculates the sequence and its associated probabilities, making it suitable for handling complex data structures.(Settles 2009)

However, basic uncertainty sampling sometimes overlooks valuable information from the probability distribution of other potential predictions. Variants such as margin sampling (Scheffer *et al.* 2001) and entropy-based (Shannon 1948) uncertainty sampling have been introduced to address this issue, considering a more holistic view of the model's predictions.(Settles 2009)

Other information-based methods include Query-by-Committee, introduced by Seung *et al.* 1992, an approach in which multiple models vote on the selection of instances through their differing predictions. Expected error reduction, a decision-theoretic approach, describes a technique where the utility function assesses the likelihood of reducing the model's generalization error (Roy *et al.* 2001).(Settles 2009)

The focus of information-based QSs on identifying data points near the decision boundaries may overlook the broader input space and the overall data distribution. Consequently, information-based approaches often select multiple similar instances, leading to redundancies in the labeled data set $D_L$. Furthermore, information-based QSs are heavily dependent on initial labeled training data, which, if insufficient, can cause the model to extrapolate and thereby compromise the learning process.(Tharwat *et al.* 2023)

#### Representation-based

Representation-based QSs focus on leveraging the structure of the unlabeled data to identify data points that capture the overall input space structure. The utility function of these QSs is oriented towards evaluating the representativeness of data points in $D_U$ and aims to query those that best illustrate the overall data distribution. This approach particularly stands out from information-based QSs when the labeled data set $D_L$ is small, such as in the early stages of AL, where it aims to enhance the learner's exploratory capabilities.(Tharwat *et al.* 2023)

Kumar *et al.* 2020 as well as Tharwat *et al.* 2023 categorizes representation-based QSs into density-based and diversity-based QSs. Motivated by applications in parallel environments, where querying data points from different sources may lead to redundancy, the diversity-based approach seeks to select those data points that exhibit significant diversity compared to labeled data in $D_L$ (Xu *et al.* 2007).

In the density-based QSs, representative data points are chosen from regions of the input space that show high density, thus reflecting the overall data distribution. The utility functions employ various metrics to assess representativeness, typically analyzing the distances between feature vectors to determine proximity.(Tharwat *et al.* 2023)

One density-based QS introduced by Ebert *et al.* 2012 utilizes a graph density method to identify densely connected nodes within the input space. This approach begins by constructing a graph with $k$-nearest neighbors where a connection, or edge, exists between nodes $x_i$ and $x_j$ if the distance $d(x_i, x_j)$ between them ranks among the $k$ smallest distances for $x_i$, using the Manhattan distance metric and a set $k$-value of 10. The edges of this graph are symmetric and weighted with a Gaussian kernel, expressed as

$$W_{ij} = P_{ij} \exp\left(-\frac{d(x_i, x_j)}{2\sigma^2}\right), \tag{2.22}$$

where $P_{ij}$ indicates the presence of an edge between $x_i$ and $x_j$. The weight matrix derived from this formula is crucial for assessing the representativeness of data points as it ranks them based on their weighted connections within the graph structure. To normalize these weights and thus prevent the repeated selection of nodes from the same dense regions, the graph density for each node $x_i$ is calculated as the ratio of the sum of the weights of the connected edges to the sum of their corresponding binary indicators.

Tharwat *et al.* 2023 lists cluster-based approaches as a third type of representation-based QS. The utility function in these approaches evaluates the selection criterion by clustering the input space using clustering techniques and selecting the nearest instances around the cluster centers (Ienco *et al.* 2013).

One such clustering method is the $k$-means algorithm, which Zhdanov 2019 utilizes in a QS to sample instances. The k-means algorithm organizes the dataset into a predetermined number of clusters ($k$), assigning each data object to the nearest cluster center. This is achieved by minimizing the sum of squared distances between data points and their respective cluster centers, given by

$$S = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{2.23}$$

where $C_i$ represents the set of points assigned to cluster $i$, and $\mu_i$ is the center of cluster $i$. The algorithm begins with an initial assignment of cluster centers, often chosen randomly, and then iterates through two main steps: (1) assigning each data point to the nearest cluster center and (2) updating the cluster centers to the mean of the points assigned to them. These steps are repeated until the positions of the cluster centers no longer change significantly, indicating that the clusters have stabilized.(Zhdanov 2019)

Representation-based QS are particularly adept due to their ability to select instances that represent the structure of the input space, thus facilitating outlier detection. Moreover, they prevent sample bias and the selection of redundant instances. However, representation-based QS tend to require more queries to cover relevant and uncertain areas effectively.(Tharwat *et al.* 2023)

## 2.2 Trustworthy Artificial Intelligence

The previous sections presented the technical fundamentals from the fields of ML and AL. These disciplines focus on developing models capable of learning from data and making decisions based on that information. Such models are increasingly playing a central role in many application areas. They are used in AI systems whose decision-making processes have consequences for humans who interact with or are affected by them. The following section aims to provide a concise introduction to the theoretical and conceptual foundations that underpin trustworthy AI, a vast and multifaceted field, reflecting these technologies' complexity and broad applicability.

Against the backdrop of the growing integration of these technologies into everyday life, the concept of trustworthy AI is gaining increasing importance. This concept covers the performance as well as the aspects of the trustworthiness of such systems. To this end, numerous proposals for AI principles have been published in the past (Toreini *et al.* 2020) from which various frameworks have been formed that, regardless of their terminology of responsible or ethical AI, address technical, legal, and social implications.(Thiebes *et al.* 2021)

In 2019, the independent High-Level Expert Group on AI of the European Commission introduced its ethics guidelines for trustworthy AI (High-Level Expert Group on AI 2019). These guidelines have quickly become established and form the basis for integrating the concept of trustworthy AI into further guidelines and frameworks, such as the OECD (Organisation for Economic Co-operation and Development) Principles for AI (OECD 2024).(Thiebes *et al.* 2021)

The authors (Thiebes *et al.* 2021) delve into the broader definition of trustworthy AI[1], and follow the High-Level Expert Group on AI 2019, proposing that

> *„ [...] AI is perceived as trustworthy by its users (e.g., consumers, organizations, society) when it is developed, deployed, and used in ways that not only ensure its compliance with all relevant laws and its robustness but especially its adherence to general ethical principles."*

The guidelines have found broad acceptance in research and practice (Hickman *et al.* 2021; Liu *et al.* 2023; Kaur *et al.* 2021; Kaur *et al.* 2023), forming the core of the framework depicted in Figure 2.8. This framework is built on three components that are essential for trustworthy AI throughout its entire life cycle (High-Level Expert Group on AI 2019):

- **Lawful AI:** AI systems should operate within the bounds of existing laws, including European, national, and international regulations.
- **Ethical AI**: AI systems should be aligned with ethical norms to ensure their operation and outcomes are morally sound and justifiable.
- **Robust AI:** AI systems should function safely and reliably in both technical and social contexts.

The foundations for trustworthy AI according to High-Level Expert Group on AI 2019 are established by building on fundamental rights and

[1] The definition according to ISO/IEC TR 24028:2020 2020 for trustwothy AI is: *„Trustworthy AI is a framework to ensure that a system is worthy of being trusted based on the evidence concerning its stated requirements. It makes sure that the users' and stakeholders' expectations are met in a verifiable way."*

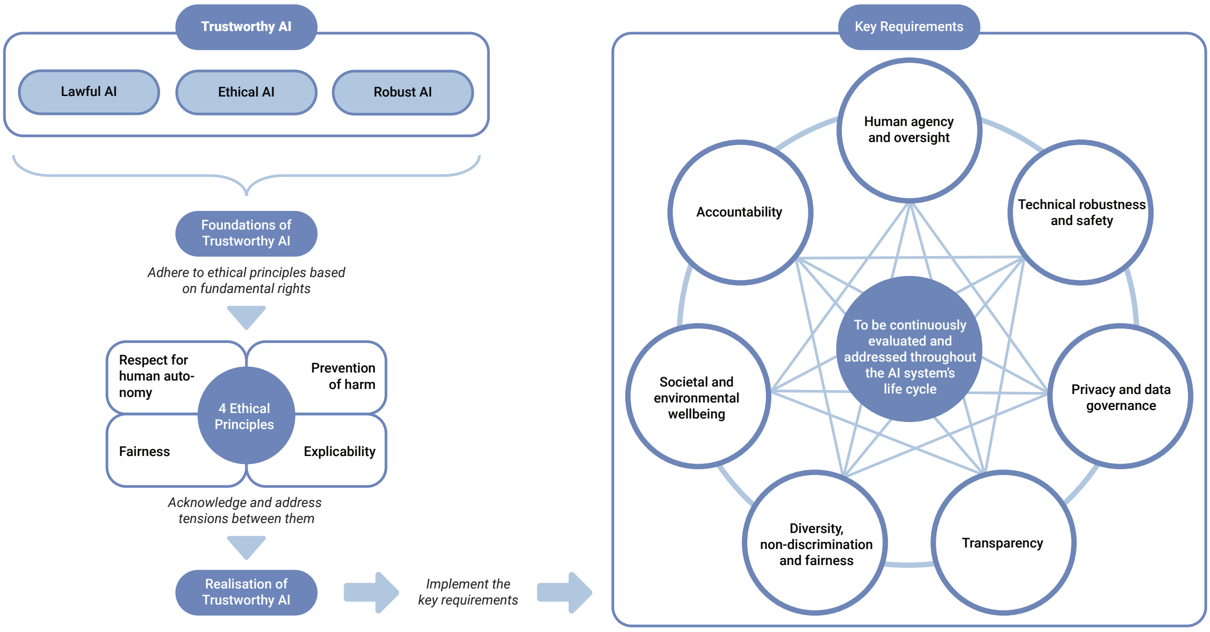

Trustworthy AI
Lawful AI
Ethical AI
Robust AI
Foundations of Trustworthy AI
Adhere to ethical principles based on fundamental rights
Respect for human autonomy
Prevention of harm
4 Ethical Principles
Fairness
Explicability
Acknowledge and address tensions between them
Realisation of Trustworthy AI
Implement the key requirements
Key Requirements
Human agency and oversight
Technical robustness and safety
Privacy and data governance
Transparency
Diversity, non-discrimination and fairness
Societal and environmental wellbeing
Accountability
To be continuously evaluated and addressed throughout the AI system's life cycle

**Figure 2.8:** High-level framework for trustworthy AI according to High-Level Expert Group on AI 2019: The three main components are incorporated into the foundations for trustworthy AI, which set out an approach based on fundamental rights. The four ethical principles (left) are translated into seven key requirements (right). These key requirements are considered equally important and interdependent – they rely on and support each other. The goal of trustworthy AI is to ensure that these seven key requirements are implemented and evaluated throughout the entire life cycle of the AI system.

summarizing four ethical principles. These principles originate from the field of ethics, specifically AI ethics - a subfield of applied ethics that deals with the ethical challenges arising from AI development, deployment, and use (Floridi *et al.* 2018). The primary focus is determining how AI can enhance or impair the quality of life, human autonomy, and the freedoms necessary for a democratic society.

To realize trustworthy AI, the guidelines articulate seven high-level key requirements that address the four ethical principles, which are briefly introduced in the following section.

## 2.2.1 Ethical Principles

The ethical principles are designed to ensure that AI systems operate within legal parameters and uphold and promote ethical standards essential for societies. These principles aim to ensure that AI systems contribute to individual and collective well-being and promote a fair and just society. The ethical principles are designed to evolve alongside the socio-technical environment, influencing both the creation of regulatory instruments, such as the approved EU AI Act (European Commission 2024), as well as the interpretation of fundamental rights. According to High-Level Expert Group on AI 2019, they include:

- **Respect for Human Autonomy:** AI systems should enhance human agency and decision-making capabilities without creating dependencies. This principle requires that AI respects human dignity and supports our ability to make decisions freely and without undue influence, ensuring that individuals maintain control over their personal and societal interactions.

- **Prevention of Harm:** AI systems must operate in a way that protects users and society from harm. This includes safeguarding physical and mental integrity, personal data, and privacy. In the development and operation of AI, safety and robustness should be prioritized to actively prevent harm wherever possible and to address potential vulnerabilities to mitigate risks.
- **Fairness:** Fairness must be central to AI development to ensure that AI systems do not perpetuate existing biases or create new forms of discrimination. This involves equitable treatment of all users and the just distribution of AI's benefits and burdens. AI should facilitate access to opportunities and resources in a way that promotes equality and non-discrimination.
- **Explicability:**[2] Transparency and understandability are crucial for maintaining user trust and enabling accountability. AI systems should be understandable to their users, providing clear, comprehensible explanations for decisions where possible. When decisions are complex, it becomes even more critical that the processes behind them are transparent and subject to oversight.

Although the concepts in this dissertation primarily touch on the principle of explicability, it is important to consider this in the context of all four principles since they dynamically interact within AI applications. While they guide the ethical development and implementation of AI systems, tensions may arise between them, requiring careful consideration and balance. Often, contextual judgment is necessary to effectively manage conflicts, such as between individual privacy and public security or between enhancing autonomy and preventing harm.(High-Level Expert Group on AI 2019)

### 2.2.2 Key Requirements

Trustworthy AI demands that ethical principles be transformed into concrete requirements across the life cycle of AI systems, involving various stakeholders such as developers, deployers, end-users, and the broader society. According to High-Level Expert Group on AI 2019 these stakeholders each play a separate role in ensuring that these requirements are met. For example, developers must incorporate these requirements into the design and development of AI systems; deployers (provider) must ensure their systems and services meet these standards; end-users and the broader society should be informed and able to demand adherence to these requirements.

In context of this thesis, while the primary focus is on the key requirement of transparency, it is essential to recognize that this requirement, like others, is interconnected and influences the overall framework for trustworthy AI as depicted in Figure 2.8. The key requirements proposed in the guidelines (High-Level Expert Group on AI 2019) have mutual dependencies and influence each other, necessitating a brief introduction to all seven key requirements set forth for trustworthy AI:

- **Human Agency and Oversight:** This requirement emphasizes supporting human autonomy in interactions with AI systems,

[2] As stated in the guidelines, „*Explicability is crucial for building and maintaining users' trust in AI systems. This means that processes need to be transparent, the capabilities and purpose of AI systems openly communicated, and decisions – to the extent possible – explainable to those directly and indirectly affected. Without such information, a decision cannot be duly contested. An explanation as to why a model has generated a particular output or decision (and what combination of input factors contributed to that) is not always possible. These cases are referred to as 'black box' algorithms and require special attention. In those circumstances, other explicability measures (e.g., traceability, auditability, and transparent communication on system capabilities) may be required, provided that the system as a whole respects fundamental rights. The degree to which explicability is needed is highly dependent on the context and the severity of the consequences if that output is erroneous or otherwise inaccurate.*"(High-Level Expert Group on AI 2019)

preventing over-reliance on technology. It ensures that AI systems augment rather than replace human decision-making, maintaining a balance where humans retain control over, and responsibility for, AI-driven decisions.

- **Technical Robustness and Safety:** AI systems must be resilient and operate reliably under a variety of conditions. This involves ensuring AI systems are free from vulnerabilities that could lead to failures or accidents, thus safeguarding users and the environment from potential harm. The systems must include robust security measures to prevent malicious attacks and ensure data integrity.
- **Privacy and Data Governance:** Effective management of data is crucial. This involves ensuring that personal data collected by AI systems is processed securely and privately, respecting user confidentiality throughout the system's life cycle. Proper data governance further means that data is accurate, processed lawfully, and used ethically, aligning with established data protection regulations.
- **Transparency:**[3] The operations within AI systems should be transparent, making it possible for users to understand and trust the technology. This includes ensuring that the decisions made by AI are explainable[4] and that the processes leading to those decisions are accessible and understandable. Furthermore, transparency involves clear communication about the AI system's capabilities and limitations.
- **Diversity, Non-Discrimination, and Fairness:** AI systems should be designed and operated in a manner that prevents bias and ensures fairness and inclusivity. This means actively addressing potential biases in AI programming and data sets, and ensuring that AI applications do not discriminate against any individual or group.
- **Societal and Environmental Well-being:** AI should contribute positively to societal goals and operate sustainably. This requirement focuses on the broader impact of AI technologies, encouraging solutions that enhance societal well-being and operate without harming the environment.
- **Accountability:** There must be clear mechanisms in place to hold AI systems and their operators accountable for their performance and impacts. This includes having clear processes for auditing and monitoring AI systems and for addressing any issues or harms that arise. Accountability ensures that those affected by AI systems have avenues for redress and that AI operators can provide justifications for decisions and actions.

The relationship between the requirements for trustworthy AI is exemplified in the survey by Ge *et al.* 2024, where they are juxtaposed in a matrix. In the review by Li *et al.* 2023, representative explicit interactions among aspects of AI trustworthiness are graphically depicted in dependency, with the authors distinguishing between technical, ethical, and other requirements and categorizing relationships into „Trade-off with“, „contribute to“, and „manifest in“.

This dissertation emphasizes the key requirement of transparency related to the other requirements. Regarding controlling and understanding AI Systems, these requirements enable users to understand and effectively

[3] Transparency in the context of AI systems is defined in ISO/IEC TR 29119-11:2020 2020 to a „*Level of accessibility to the algorithm and data used by the AI-based system*“. In ISO/IEC TR 24028:2020 2020 the definition is „*Transparency of AI systems relates to making the data, features,algorithms, training methods and quality assurance processes available to external inspection by a stakeholder.*“

[4] Explainability in the context of AI systems is defined in ISO/IEC TR 29119-11:2020 2020 to a „*Level of understanding how the AI-based system came up with a given result*“. An alternative definition is given by ISO/IEC TR 24028:2020 2020, which states that „*Explainable AI systems would aim to provide an understanding of the processes contributing to the truth, accuracy and reasonableness of its results beyond the inductive observation that the systems seem to work.*“

monitor the operations of AI, which is indispensable for maintaining human agency and oversight (Floridi 2021; Miller 2019). Moreover, they are relevant for technical robustness and safety as they facilitate a well-founded evaluation as well as validation of the reliability and security of AI systems (Ali *et al.* 2023).

Transparency contributes to adhering to data protection and data governance standards. A profound understanding of data management processes fosters trust and ensures compliance with legal regulations. Furthermore, this requirement promotes fairness and helps prevent discrimination by enabling precise analysis and correction of potential biases in AI operations, thus ensuring equitable treatment of all user groups (Zhou *et al.* 2022; Dey *et al.* 2022).

Moreover, AI's societal and environmental impacts significantly benefit from transparency, enabling stakeholders to evaluate technologies and align them with societal goals and sustainable practices (Hagendorff 2020). The co-dependency between transparency, explainability, and accountability is evident as transparent processes and systems enable monitoring and audits, allowing developers and providers to be held accountable for their performance and impacts, thus fulfilling accountability requirements.(High-Level Expert Group on AI 2019)

## 2.3 Explainable Artificial Intelligence

In the concepts of trustworthy AI, a crucial aspect is that systems are transparent. The field of XAI addresses this need by contributing to improving the explainability of AI systems. XAI aimed to make the decision-making processes of AI models more understandable to humans. This section focuses on the computer science perspective of this field[5] and elaborates on the methods of IML relevant in this dissertation.

Vilone *et al.* 2021 provide a summary of definitions of notations related to the concept of explainability, which they compiled from an extensive literature review. They found that the term explainability is often replaced by the term interpretability and is considered synonymous within the research community. For this reason, the terms are treated as such in this thesis.

Nevertheless, due to the significant attention this research field has received in recent years, various terminologies according XAI and IML have been established. Schwalbe *et al.* 2023 defines XAI as

> *„ [...] the area of research concerned with explaining an AI system's decision."*

Molnar 2022, in his often-cited eponymous work, refers to the field of IML as

> *„ [...] methods and models that make the behavior and predictions of machine learning systems understandable to humans."*

### 2.3.1 Interpretable Machine Learning

Driven by the increasing complexity and opacity of ML models, which, despite their high predictive performance, often offer little insight into the underlying decision-making processes, there is a push to enhance the comprehensibility of these systems for various users through suitable interpretation mechanisms. Given the rapidly growing research field, numerous comprehensive reviews exist in the literature and taxonomy proposals of this research field (Arrieta *et al.* 2020; Arya *et al.* 2019; Carvalho *et al.* 2019; Schwalbe *et al.* 2023; Saeed *et al.* 2023; Speith 2022).

A central topic in technical XAI research is the balance between the performance of ML models, such as predictive accuracy, speed, and resource consumption, and their explainability. In simplified terms, more complex models often promise higher predictive accuracy but tend to be less transparent and, thus, harder to interpret. IML is relevant in this context since it technically contributes as the basis for many XAI concepts. Due to the plethora of methods published in recent years, there are extensive compilations on this (Adadi *et al.* 2018; Molnar 2022).

Figure 2.9 illustrates a simplified version of the taxonomy proposed by Speith 2022, in which the methods are categorized along various dimensions. Typically, approaches are differentiated along the stage dimension between ante-hoc, intrinsically explainable models, corresponding to the Explainable-by-Model Design paradigm (Rudin *et al.* 2022), and post-hoc.

[5] Some text sections are from contributions published in Section *„XAI in Computer Science"* in Ziethmann *et al.* 2024

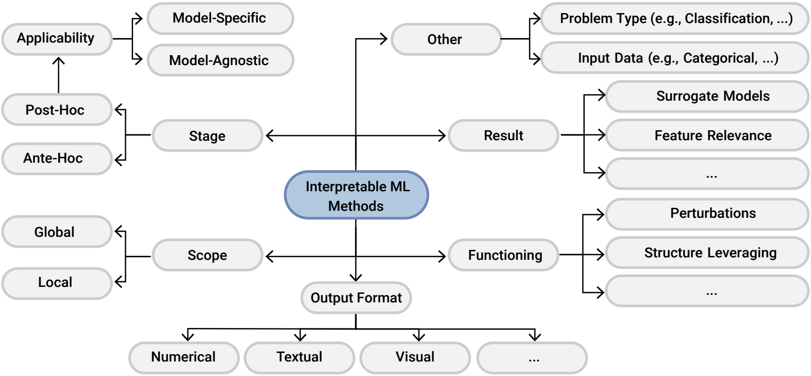


**Figure 2.9:** Simplified taxonomy of IML methods based on Speith 2022.

The latter includes methods to provide insights into a black-box model's workings after being trained.

In addition to these stage-based distinctions, methods for IML can be categorized based on the results they produce. This includes techniques such as surrogate models and feature relevance analysis. Surrogate models approximate the behavior of complex models with simpler, interpretable ones, like linear regression, to provide an overall understanding of model predictions. Feature relevance methods assess the importance of individual features in the model's decision-making process, identifying which inputs significantly impact the predictions.

Techniques like Local Interpretable Model-agnostic Explanations (LIME, proposed by Ribeiro *et al.* 2016), and Shapley Additive Explanations (SHAP, proposed by Lundberg *et al.* 2017) are prominent examples, which will be detailed in Section 2.3.2. These methods are suitable for different model types and are generally characterized as model-agnostic. Additionally, there are methods for specific model architectures, such as CNNs, which create, for instance, Pixel Attribution Saliency Maps, including Gradient-weighted Class Activation Mapping (Grad-CAM, proposed by Selvaraju *et al.* 2017). Further differences arise regarding supported input data formats; for instance, Randomized Input Sampling for Explanation (RISE, proposed by Petsiuk *et al.* 2018) is model-agnostic but is suitable only for image data.

Another possible categorization is according to the scope of the explanation, whether it encompasses individual predictions (local) or the entire model (global). Global model interpretability involves comprehending how a model makes predictions by examining all features and learned components like weights. However, this is challenging for complex models with numerous parameters, as human cognitive limits make it difficult to visualize multi-dimensional feature spaces. Consequently, interpretability often requires focusing on specific model aspects, such as weights in linear models. Modular interpretability examines how model parts contribute to predictions, offering insights into elements like linear model weights, which require context for accurate interpretation.(Molnar 2022)

Local interpretability, on the other hand, focuses on specific predictions, providing detailed explanations for individual instances. This can reveal simpler dependencies than global interpretations, offering more precise insights. Explaining predictions for a group of instances can be done using global or local methods, either treating the group as a complete dataset or aggregating individual explanations to understand the model's behavior for that subset.(Molnar 2022)

The results computed by the methods can often result in statistical metrics, which technical experts primarily use. To address this dilemma, another part of technical XAI research explicitly deals with the stakeholder perspective (Gleicher 2016; Langer *et al.* 2021). This area, for example, involves visualization techniques aimed at translating complex model decisions into visually interpretable formats. Besides textual or numerical output formats, heatmaps, for instance, are used to highlight which parts of an image sample are crucial for the model's classification.

Further, XAI research is viewed through the lens of HCI (Ferreira *et al.* 2020; Miller 2019; Mueller *et al.* 2021). In this domain, the concept of interactive explanations is gaining importance. These approaches allow users to interact with the AI system through dialogues and feedback loops, fostering an understanding of the model outputs (Chromik *et al.* 2020).

Generating explanations using IML aims, as mentioned above, to make the model's decision-making processes more understandable for humans. Social sciences can help determine what constitutes a „good" explanation, where Miller 2019 has provided a comprehensive overview. Molnar 2022 summarizes this by noting that people prefer brief explanations. Explanations are social interactions, which is why the social context significantly influences the content of the explanation (Miller 2019).In this context, evaluating the explanations through appropriate methods and measurable metrics is crucial in assessing the effectiveness and usefulness of various approaches.

Typically, the understandability, relevance, and accuracy of the explanation are evaluated (Nauta *et al.* 2023), which, according to the approach of Doshi-Velez *et al.* 2017, can be determined concerning specific tasks and either through human or non-human input. They propose three main levels of evaluation:

- **Application Level Evaluation:** Explanations are tested in real-world scenarios with end users. For example, in a customer service application, support agents might use a chatbot to determine how well its explanations help resolve customer inquiries, comparing the explanations to traditional support methods.
- **Human Level Evaluation:** Involves testing explanations with laypersons rather than experts, making it more cost-effective. Participants choose the best explanations, providing insights into their clarity and effectiveness for non-experts.
- **Function Level Evaluation:** Evaluations are conducted without human input, using proxies like model complexity to assess interpretability. For example, decision tree depth might serve as a proxy, with shallower trees indicating better interpretability, provided predictive performance remains strong (Molnar 2022).

### 2.3.2 Methods for Local Post-hoc Explanations

There is a wide range IML methods in the literature. The following section introduces the methods relevant to this dissertation, which focus on the post-hoc approach and are primarily used to generate local explanations.

#### Randomized Input Sampling for Explanation (RISE)

The core idea of RISE, introduced by Petsiuk *et al.* 2018, is to assess the importance of different parts of an input by observing how random perturbations affect the model's output. This method involves creating a series of randomly masked versions of an input image and analyzing the classifiers predictions for each masked input. This approach provides insight into which regions of the input are most influential in the model's decision-making process.

The algorithm involves generating a series of $N$ random binary masks, $\{M_1, M_2, ..., M_N\}$, where each element of a mask is set to 1 (showing the pixel) with a certain probability and 0 otherwise. These masks are scaled up to the input's dimensions and used to perturb the input image by element-wise multiplication.

To evaluate the importance of each pixel $\lambda$, RISE computes the expected model output when the pixel is visible, denoted by $S_{I,f}(\lambda)$. This score is calculated by averaging the outputs across all perturbed images where the pixel $\lambda$ is unmasked, as described by:

$$S_{I,f}(\lambda) = \frac{1}{E[M]} \sum_{m} f(I \odot m) \cdot m(\lambda) \cdot P(m = M) \tag{2.24}$$

where $f(I \odot m)$ is the model's output for the perturbed image, $m(\lambda)$ indicates whether pixel $\lambda$ is visible in mask $m$, and $P(m = M)$ represents the probability of mask $m$ being applied. This method assigns more weight to the masks that reveal the pixel, hence providing insights into which pixels significantly impact the model's decision-making process. By computing the relevance scores for all pixels, RISE generates a saliency map that visualizes the influential regions of the input image. (Petsiuk *et al.* 2018)

#### Gradient-weighted Class Activation Mapping (Grad-CAM)

Grad-CAM, developed by Selvaraju *et al.* 2017, is a prominent gradient-based method used to generate saliency maps within the category of post-hoc IML approaches. These method calculate the model's output gradients with respect to input features to assess their impact on decision-making. This contrasts with other gradient-based techniques like vanilla gradients (Simonyan *et al.* 2014) and guided backpropagation (Springenberg *et al.* 2015), which vary in their backpropagation algorithm handling.

The essence of Grad-CAM lies in its generation of a coarse localization map that identifies crucial image regions for class prediction. This process begins by computing the gradients of the model's output with respect to the feature maps in the last convolutional layer. These gradients reflect each feature map's influence on the final class output.

The crucial computation in Grad-CAM involves assigning weights to these feature maps. These weights, $\alpha_k^c$, are derived by globally averaging the gradients across all spatial locations within each feature map, producing a scalar that quantifies each map's importance for the class $c$:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k^c \cdot A^k\right) \tag{2.25}$$

Here, $A^k$ are the feature maps, and $\alpha_k^c$ are the weights calculated as described. Combining these weights with their respective feature maps and applying a ReLU (2.16) function filters out non-positive influences, creating the resulting heatmap. (Selvaraju *et al.* 2017)

### Local Interpretable Model-agnostic Explanations (LIME)

LIME, a post-hoc model-agnostic method proposed by Ribeiro *et al.* 2016, is independent of the input data type and is based on a perturbation-based approach. The underlying method of LIME starts with selecting an instance for which an explanation is sought. Subsequently, a dataset with perturbed samples in the vicinity of this example is created, and the original model is used to predict the outcomes for these variations. These predictions are then fed into a simpler, interpretable model – typically a linear regression model – that has been trained to approximate the behavior of the black-box model locally around the selected instance.

The simplicity of this surrogate model ensures that its operation is understandable to humans, thereby making the functioning of the black-box model more comprehensible for the selected local instance. In other words, the interpretability of the explanation depends on the simplicity of the model used. The key is to find a balance between fidelity to the original black-box model and the inherent transparency of the surrogate model to ensure that the predictions generated are both accurate and comprehensible to humans.

To quantify the significance of features in influencing the model's prediction for a particular class, a LIME-generated explanation $\xi$ for the instance $x$ is given by:

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.26}$$

where $f$ denotes the complex original model and $g$ the simpler interpretable model. Here, $\mathcal{L}$ measures how well $g$ approximates $f$ near the instance, with $\pi_x$ expressing the weights or the locality around $x$, and $\Omega$ is a regularization term that controls the complexity of model $g$.

The quality and usefulness of the explanations generated by LIME can vary depending on the perturbation strategy. Moreover, generating explanations requires a high computational effort, especially for complex, multidimensional datasets. Another methodological aspect is the locality of the explanations. LIME generates explanations only in the direct vicinity of a selected instance, with the captured feature dependencies being specific for the proximity of the respective instance and not necessarily universally valid for the entire model behavior. This local focus can lead to different explanations for similar examples, depending on where and how the perturbations are performed. Conversely, this characteristic is advantageous for examining the robustness of the generated explanations.(Ribeiro *et al.* 2016)

### Shapley Additive Explanations (SHAP)

SHAP, devised by Lundberg *et al.* 2017, employs the game-theoretically optimal Shapley values to elucidate individual model predictions. Each feature in a dataset is treated as a „player" in a game where the model's prediction is the payoff. SHAP leverages this analogy to fairly distribute the prediction „payout" among all features, reflecting each one's contribution to the outcome. (Molnar 2022)

Central to SHAP is the computation of Shapley values (Shapley 1953). Specifically, it calculates the marginal contribution of each feature across all possible combinations, or coalitions, of features. Lundberg *et al.* 2017 defined an explanation as

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{2.27}$$

Here, $g$ represents the explanation model, $z' \in 0, 1^M$ is the coalition vector indicating the presence (1) or absence (0) of features in the coalition, $M$ is the total number of features, and $\phi_j$ are the Shapley values or feature attributions. This linear representation aligns with the principles of additive feature attribution, similar to LIME but extended through the lens of game theory.

The concept of Shapley values in SHAP centers around the idea that the contribution of each feature to the prediction can be calculated by simulating the inclusion or exclusion of that feature in various combinations of other features. The Shapley value for a feature essentially represents its average marginal contribution across all possible subsets of features, ensuring that each feature's contribution to the prediction is measured fairly and consistently.(Lundberg *et al.* 2017)

Moreover, SHAP transcends its role as merely a local explanation tool; it incorporates global interpretation methods that aggregate Shapley values to provide broader insights into model behavior. This global perspective enables a deeper understanding of model predictions across the entire dataset, not just at the individual prediction level. (Molnar 2022)

## 2.4 Software Engineering for AI Systems

The development of AI systems involves unique characteristics that require specialized approaches from the field of software development. Unlike traditional software systems, AI systems are more data-driven and probabilistic. In traditional software development, individual components or entire systems are designed through a series of steps, from requirements to final implementation, organized into stages and phases known as the development process. Common approaches include sequentially (e.g., the waterfall model, Benington 1983), iterative (e.g., the V-model, Forsberg *et al.* 1991) and agile (e.g., Scrum, Schwaber 1997) approaches, which differ in their structuring of procedures.

In AI systems, complexity arises from the dependency on the algorithms and models being developed. Regardless of whether the models are trained using supervised or unsupervised methods, they heavily rely on large datasets. Unlike traditional software development projects, data-intensive projects pose particular challenges due to the inherent uncertainty regarding the quality of the solutions. Consequently, process models from conventional software development are often inadequate, as they do not sufficiently account for the data-intensive aspects. To address this, data science process models have been established that cover the exploratory nature of these projects and incorporate these aspects within a development process.(Kutzias *et al.* 2023)

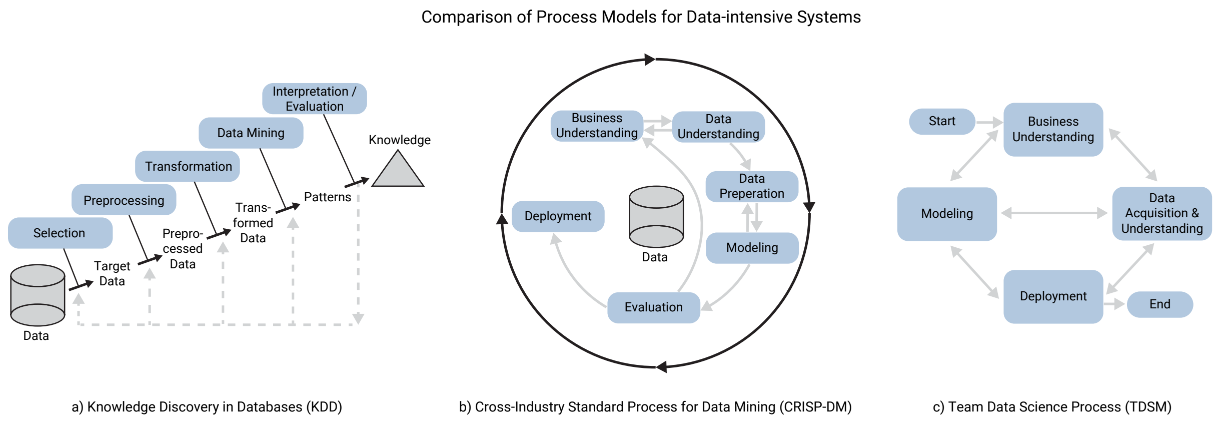
### 2.4.1 Process Models for Data-intensive Systems

In the following, three prominent process models for data-intensive systems will be presented.[6]

**Knowledge Discovery in Databases (KDD)** One of the earliest process models originates from the data mining practice known as KDD, proposed by Fayyad *et al.* 1996. KDD is defined as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. The structure of KDD is illustrated in Figure 2.10(a), which describes the five main phases and the flow between them.

The process begins with the selection phase, in which relevant data is identified and extracted from a larger data set. The selected data, referred to as Target Data in the diagram, is then transferred to the preprocessing phase, resulting in Preprocessed Data. This is followed by the transformation phase, where the data is converted into formats suitable for mining, often involving normalization or aggregation. In the data mining phase, algorithms are applied to extract patterns from the processed data. Finally, in the interpretation/evaluation phase, the patterns are analyzed to determine their significance and usefulness. The five main phases in KDD are designed to be interactive and iterative, as indicated by the feedback loops, and include several decision-making steps to achieve optimal results.(Fayyad *et al.* 1996)

[6] Edited text from the publication Stieler *et al.* 2024



a) Knowledge Discovery in Databases (KDD)

b) Cross-Industry Standard Process for Data Mining (CRISP-DM)

c) Team Data Science Process (TDSM)

**Figure 2.10:** Comparison of process models for data-intensive systems, illustrating the distinct methodologies and workflow stages. Panel (a) displays the Knowledge Discovery in Databases (KDD) process (Fayyad *et al.* 1996), emphasizing the sequential steps from data selection to knowledge interpretation. Panel (b) outlines the Cross-Industry Standard Process for Data Mining (CRISP-DM), proposed by Wirth *et al.* 2000, highlighting its cyclical, iterative approach from business understanding to deployment. Panel (c) represents the Team Data Science Process (TDSP), developed by Microsoft 2020, showing a structured, team-oriented workflow that integrates data acquisition and understanding through to deployment.

**Cross-Industry Standard Process for Data Mining (CRISP-DM)** CRISP-DM, introduced by Wirth *et al.* 2000, is another well-established process model, comprising six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Like KDD, CRISP-DM is iterative, accommodating the often nonlinear nature of data mining projects – both process models are compared in the survey of Mariscal *et al.* 2010.

The phases as well as the CRISP-DM workflow are illustrated in Figure 2.10(b). The process starts with the Business Understanding phase, which ensures that the project aligns with the organization's objectives. It involves defining the project requirements from a business perspective and converting this knowledge into a data mining problem definition. The subsequent Data Understanding phase involves collecting initial data, familiarizing oneself with the data, and identifying any data quality issues that need to be addressed. In the Data Preparation phase, data is selected, cleaned, and formatted into a final dataset for modeling. During the Modeling phase, various modeling techniques are chosen and applied to the prepared dataset.(Wirth *et al.* 2000)

During the Evaluation phase, models are assessed to ensure that they meet the business objectives set out at the beginning of the project. The feedback loops shown in the CRISP-DM diagram illustrate the circular design of CRISP-DM. Particularly from the evaluation phase back to the business understanding phase, the business goal-oriented focus of this process model is underlined. The final phase, Deployment, involves applying the model in a practical setting, which could include generating reports or integrating the model into an operational system.(Wirth *et al.* 2000)

**Team Data Science Process (TDSP)** In contrast to traditional process models for data mining projects, the TDSP, proposed by Microsoft 2020, represents a significant shift towards modern data science practices. TDSP is further iterative but envisages a more agile procedure, enabling effective and efficient collaboration within data science teams. It breaks down the data science process into five clearly defined phases: Business Understanding, Data Acquisition and Understanding, Modeling, Deployment, and Customer Acceptance.

Each phase includes specific tasks and milestones and assigns clear roles to individual team members to increase the team's productivity through high transparency and coordination. The Business Understanding phase sets the stage for the project, similar to CRISP-DM, but with a more agile approach that allows for iterative feedback and adjustments. The Data Acquisition and Understanding phase focuses on gathering the necessary data and understanding its structure and quality, which is essential for building reliable models.(Microsoft 2020)

The Modeling phase involves developing predictive models, emphasizing rapid prototyping and iteration to refine the models. The Deployment phase in TDSP is more integrated with the operational aspects of the organization, ensuring that models are deployed, monitored, and maintained over time. Finally, the Customer Acceptance, illustrated by the arrow between Deployment and End, includes system validation, which confirms that the deployed model and pipeline meet the customer's requirements, and project hand-off, which specifies the step at which the project is delivered to the person or team who will use the system in production.(Microsoft 2020)

**Further Approaches.** Additional process models have been introduced, including the Analytics Solutions Unified Method (ASUM) (IBM Corporation 2016), the lightweight IBM Cloud Garage Method for data science (ILG) (Kienzler 2020), Engineering Data-Driven Application (EDDA) (Hesenius *et al.* 2019), and the Data Science Process Model (DASC-PM) (Schulz *et al.* 2020), which are compared in the work of Kutzias *et al.* 2023.

Another modern concept focusing on ML/AI projects was presented by Studer *et al.* 2021. The authors propose the Cross-Industry Standard Process Model for the Development of Machine Learning Applications with Quality Assurance Methodology (CRISP-ML(Q)) which is compatible with CRISP-DM. The main difference from CRISP-DM is that the „Business-" and „Data Understanding" phases are combined, and a „Monitoring & Maintenance" phase complements the overall process.

### 2.4.2 Operation Frameworks

Following the introduction of process models that focus on the structured development of data-intensive systems, an important consideration arises regarding how these concepts can be integrated into an agile operationalization framework. This is where modern approaches such as DevOps, DataOps, and MLOps come into play, offering an evolutionary response to the dynamic demands of contemporary environments and will be briefly introduced in the following.[7]

[7] Edited text from the publication Stieler *et al.* 2024

**DevOps.** DevOps, named for its fusion of „development“ and „operations“, is an integrated approach that merges software development and IT operations practices to enhance the speed and efficiency of software delivery. Developed by Debois 2008, the core concepts of DevOps recognize the need to eliminate inefficiencies in traditional software development processes, particularly addressing the separation between development and operations. Today, DevOps is a widely adopted strategy that aims to ensure continuous development and delivery throughout the entire software life cycle.(Azad *et al.* 2023)

Beyond its technical aspects, DevOps has become known for fostering collaboration between development and operations teams within organizations, as the methodologies represent a cultural shift within teams that prioritizes rapid delivery through agile and lean practices within a system-oriented framework. Despite its widespread adoption, the literature has no universally accepted definition. DevOps is best understood as a paradigm encompassing methods, principles, and practices promoting communication and collaboration.(Senapathi *et al.* 2018)

The DevOps life cycle is often depicted as an infinite loop of operations comprising seven steps: planning, coding, building, testing, releasing, deploying, operating, and monitoring. Technologies, particularly automation tools, are employed to create a programmable and dynamic infrastructure that supports the entire system life cycle.

A cornerstone of this approach is the implementation of Continuous Integration (CI) and Continuous Delivery (CD), which are essential for the rapid deployment of software updates. In CI, code changes are automatically tested and integrated into a shared repository multiple times a day. This process helps developers quickly identify and fix errors, ensuring the codebase remains stable and functional. CD, on the other hand, automates the release of tested code to production environments, allowing new features and updates to be delivered quickly and consistently.(Azad *et al.* 2023)

**DataOps** While DevOps has transformed collaboration between software development and IT operations to improve the speed and reliability of software delivery, a similar evolution is taking place in data management. As organizations increasingly rely on data-driven decision-making, flexible data operations have become crucial. This is where DataOps comes into play, offering a framework that applies DevOps principles to the entire data life cycle.(Munappy *et al.* 2020)

At its core, DataOps, like DevOps, aims to enhance the efficiency of data-intensive processes by fostering a culture of collaboration and continuous improvement among data engineers, data scientists, and IT operations teams. A key component of DataOps is the use of automated data pipelines, which facilitate data flow. These pipelines encompass steps such as data ingestion, validation, transformation, and integration, ensuring that data is accurate and ready for analysis at each stage.(Ereth 2018; Munappy *et al.* 2020)

To achieve this, concepts from DevOps, such as CI/CD, are adapted and applied to data operations. In doing so, DataOps promotes a dynamic and responsive data environment where new data products and analytics solutions can be rapidly iterated. This reduces the effort required to manage data workflows, allowing teams to focus on deriving actionable insights.

**MLOps.** MLOps extends the concepts of DevOps and DataOps to address the unique requirements of ML/AI projects. This approach harmonizes the various phases in the life cycle of AI systems and ensures efficient production, implementation, monitoring, and iteration within industrial contexts. MLOps tackles the complexity of developing and managing AI systems by providing a structured framework that integrates the efforts of data engineers, data scientists, software engineers, and ML engineers into a cohesive workflow.

MLOps aims to create a collaborative and efficient environment for AI projects, focusing on seamless integration and interaction among various roles to enhance productivity. According to Kreuzberger *et al.* 2023, MLOps incorporates principles comparable to best practices, guiding how processes should be realized. These principles include comprehensive version control for source code and data, configurations, and models. This ensures reproducibility and traceability, which are crucial for managing complex AI systems.

A key aspect of MLOps is automation, particularly in managing workflows for model training, deployment, and monitoring. MLOps integrates essential DevOps concepts like CI/CD, extending them with Continuous Training (CT), where ML models are iteratively retrained based on new data or defined triggers, such as changes in data patterns (John *et al.* 2021; Karamitsos *et al.* 2020; Lwakatare *et al.* 2020b). This approach ensures that models remain up-to-date and effective, aligning with evolving datasets and business requirements. Another fundamental principle is workflow orchestration, which coordinates tasks within ML pipelines to efficiently manage dependencies and execution order.(Kreuzberger *et al.* 2023)

The MLOps ecosystem further includes concepts related to creating a robust infrastructure that supports the entire lifecycle of AI projects. This infrastructure encompasses components for data collection, preparation, model development, validation, deployment, and subsequent continuous monitoring and maintenance.(Zhou *et al.* 2020)

# II. Development in AL Projects
# A Process Perspective

# Active Learning Development Life Cycle 3

**3.1 Project Process Model . . 58**
3.1.1 Scoping & Project Planning 58
3.1.2 Data Engineering . . . . . 59
3.1.3 Modeling . . . . . . . . . . 60
3.1.4 Maintenance . . . . . . . . 60
3.1.5 Acceptance . . . . . . . . . 61
**3.2 Inception . . . . . . . . . . 62**
3.2.1 Cold Start . . . . . . . . . . 62
3.2.2 Warm Start . . . . . . . . . 63
**3.3 Continuous Phases . . . . 65**
3.3.1 Data Iteration . . . . . . . 65
3.3.2 Machine Learning Iteration 68
3.3.3 Development Iteration . . 73
3.3.4 Operations Phase . . . . . 76
**3.4 Roles . . . . . . . . . . . . . 81**
3.4.1 Non-technical Roles . . . . 81
3.4.2 Technical Roles . . . . . . . 81
**3.5 Summary . . . . . . . . . . 83**

The effective development and integration of highly complex AI systems necessitate, irrespective of the application domain, a comprehension of the challenges accompanying implementation at both the technical and organizational levels. The foundational framework for a successful introduction and seamless deployment of AI systems is constituted by precisely formulated phases and the resultant activities for a project team, which are expounded within the context of traditional software development through so-called life cycle models.

In simple terms, a life cycle is a structured process encompassing the various steps undergone in the conception, development, and operation of a (software-) product (Machado *et al.* 2024). As delineated in Section 2.4, the substeps are sequentially and phase-based in certain process models, while others entail multiple iterations.

Regardless of this, the research community has already identified deficiencies in existing ML life cycle models (Diaz-de-Arcaya *et al.* 2023; Xie *et al.* 2021; Machado *et al.* 2024), which manifest themselves, for example, in missing phases, such as a feasibility study (Haakman *et al.* 2021), or even monitoring (Studer *et al.* 2021). Concurrently, it can be observed that most proposed life cycle models overly emphasize the aspects of the ML pipeline, inadequately integrating the phases and activities of classical software development, such as requirements management and testing (Steidl *et al.* 2023).

Transposing the characteristics of the AL methodology onto an AI project elucidates an iterative and agile approach arising due to the inherent continuity across all phases. In addition to an agile mindset, it is imperative to consider the extensions of modern ML life cycle models encompassing data centrality, model training, and exploratory tasks and integrate them into a flexible and adaptable development process. When considering the interaction of individual subtasks from the developer's perspective, both existing concepts like DevOps and emerging concepts from MLOps and DataOps are implicated in implementation.

The subsequent Chapter presents the proposition of an AL development life cycle founded upon the tenets of these three concepts. Its purpose is to synchronize established practices and the individual phases, iterations, and their interrelationships within the AL development life cycle. It is based on the publications Stieler *et al.* 2023a, in which the iterations of the AL development life cycle are described, as well as on Stieler *et al.* 2024, where the life cycle is set in a project context as well as supplemented by project dependency phases and roles.

## 3.1 Project Process Model

A myriad of actions, methods, and decisions characterize the development of AL systems. Typically, there are numerous tasks and activities to be addressed. Executing these in individual steps forms a process, which, when organized in phases, results in a process model.

Such a process model for AL projects is depicted in Figure 3.1. Within the figure, various colored circles represent the continuous phases of Data-, ML-, Development-Iteration, and the Operations Phase, with their individual stages. These stages are detailed in Chapter 3.3.

To gain a holistic understanding, let's first approach the outer circle of the figure, which defines the four overarching **conceptual main phases** of the AL project process model. These phases follow in a clockwise direction and are embedded in the context of the entire course of an AL project.

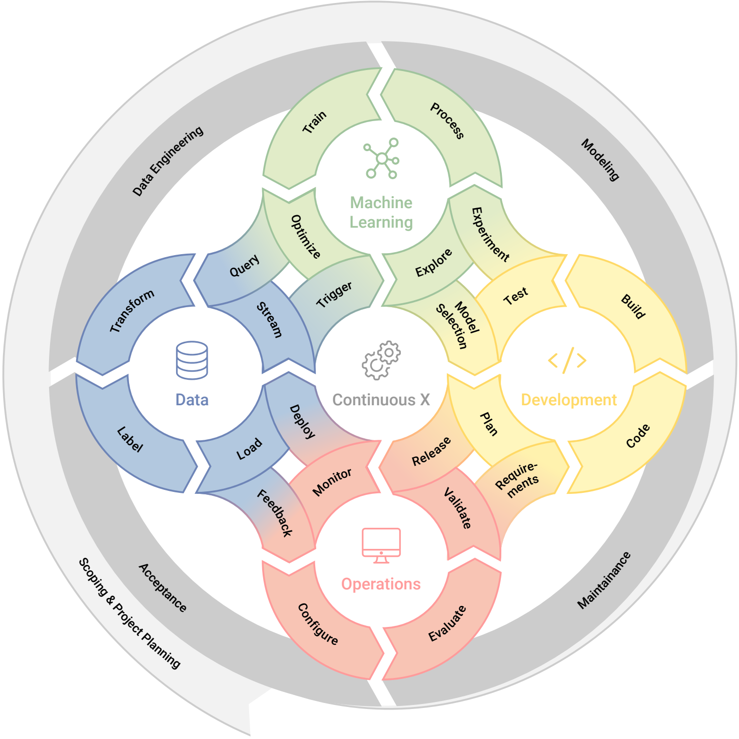


**Figure 3.1:** Active learning development process. The different colored circles represent four continuous phases of development life cycle, while the main conceptual phases of an AL project, arranged clockwise, compose the outer circle.

### 3.1.1 Scoping & Project Planning

Like other AI projects and projects in traditional software development, AL projects begin with an initial phase where various stakeholders involved define the project requirements and objectives. These are typically characterized by first identifying the specific use cases and the challenge definition that can be addressed or supported by AL. Thereby, the measurability of the requirement is a crucial principle according to the *IEEE Standard for Software Development Life Cycle* (IEEE Standards Board 1995), which contributes to the evaluation of the project (Studer *et al.* 2021).

In addition to technological considerations, the initial phase is crucial for planning the resources required for the project. These resources include financial and time budgets, as well as infrastructure components such as hardware and software. Equally important are the human resources needed for implementation, including domain experts and, in some cases, human annotators with varying qualifications as part of an AL project. Both CRISP-DM (Wirth *et al.* 2000) and *Microsoft's* TDSP (Microsoft 2020) initiate a project with the „Business Understanding" phase, during which the aforementioned tasks are performed.

*Studer et al.* suggest in their enhanced CRISP-ML(Q) merging the „Business Understanding" phase with the „Data Understanding" phase, arguing that ML projects heavily depend on data availability and quality. For this reason, the authors propose even in this initial project phase feasibility studies, which examines the applicability of ML technologies in general, legal constraints, and the requirements for the application (Studer *et al.* 2021). Although this argument seems logical, the phases in the AL development life cycle remain separated to make the process more flexible and adaptable by emphasizing the dynamic nature of data in an AL project.

As Figure 3.1 illustrates, the Scoping & Project Planning Phase further encompasses the Maintenance and Acceptance phase. This highlights the necessity of thorough project preparation that influences and supports all subsequent life cycle phases. It is evident that the Scoping & Project Planning Phase leads into the Acceptance Phase, where a comparison is made between the initially established project goals and acceptance criteria.

### 3.1.2 Data Engineering

In the initial data engineering phase, the primary focus is on comprehending and clarifying the data intended for use in the AL project or those that should be supplemented with labels. This comprehension is foundational, as subsequent process phases are built upon it.

The activities carried out in the second conceptual phase of an AL project resemble data collection. At this point, data collection predominantly refers to an inventory of the available data for the entire project or a subset serving as a representative sample (Arnold *et al.* 2020; Steidl *et al.* 2023; Tamburri 2020; Ashmore *et al.* 2022). Such data can originate from various sources and, depending on their nature - whether structured, semi-structured, or unstructured - can be held on different platforms, including data warehouses, data lakes, or data lakehouses (Roh *et al.* 2021; Zha *et al.* 2023).

Once the data is assembled, it is described based on its foundational characteristics. This includes data types, value ranges, missing values, and other preliminary statistics offering a rapid overview of the data's appearance and its current labeling status (Wirth *et al.* 2000; Studer *et al.* 2021).

An initial manual exploratory data analysis should be conducted at this stage, a procedure that will be elaborated on in more detail later in this chapter when reiterated during continuous iterations. Nevertheless, in this phase, a relevance check is carried out to filter out data that could be irrelevant or redundant for the AL task to initiate the next life cycle phase with the result of a clear understanding of the data. In more mature states of the project, additional aspects of data engineering include data management, which encompasses activities related to data versioning, data provenance, and data backup.

### 3.1.3 Modeling

The Modeling phase encompasses all tasks related to the algorithmic solution (Wirth *et al.* 2000; Microsoft 2020; Studer *et al.* 2021). Insights from the Scoping & Project Planning- and Data Engineering phases feed into this stage, forming the foundation for selecting modeling techniques. This selection is primarily dictated by the defined ML- and business objectives, the data at hand, and the project's specific constraints.

Specifically, the choice of modeling technique hinges on the nature of the task, whether it is a classification or regression problem, for instance, and the types of data available. It's inherent to the types of models that specific techniques are better suited for structured data, while others are particularly apt for unstructured data.

Therefore, not only do the project objectives defined in the Scoping & Project Planning phase, especially those concerning performance, play a role, but criteria such as complexity, interpretability, robustness, and scalability of the model influence the selection of an appropriate modeling technique. Ultimately, from the universe of all models, a subset of suitable methods emerges, deemed fitting for the AL project and considered for implementation. In this context, conducting a literature review on similar problems has established itself as a proven method to gain a comprehensive overview of the ML tasks and potential solution proposals. Published results can further serve as performance benchmarks and can be integrated into the project objectives.(Studer *et al.* 2021)

To fulfill the key characteristic of AL, adaptability must be considered when selecting the model type, to ensure an effective and efficient integration of new training data and feedback in the AL project. In this context, involving domain experts at this stage is crucial, enabling engineers to gain a comprehensive understanding of the problem domain.

Furtermore, as the project progresses, aspects of model management fall into the Modeling Phase, which includes aspects of model versioning and experiment tracking, which will be explained further in this chapter.

### 3.1.4 Maintenance

As in traditional software projects, the maintenance phase in AL/ML projects is crucial for ensuring the long-term effectiveness, reliability, and functionality of the implemented system, including the trained model. Studer *et al.* 2021 highlight this phase in their process model, arguing that ML models are often used over extended periods and have a life cycle that must be managed.

This aspect is even more pronounced in AL projects. Since ML models are regularly updated with new data or labels, key activities such as concept drift and system updates have a continuous nature during the maintenance phase. These aspects will be explicitly discussed in the operations phase later in the chapter.

### 3.1.5 Acceptance

*Microsoft's* TDSP features a final phase called the „Customer Acceptance Phase" (Microsoft 2020), which inspires the proposed process model for AL projects to merge this into an „Acceptance Phase".

On a conceptual level, evaluation, and validation refer to the project objectives defined in the Scoping & Project Planning phase. This mainly involves considering how results can be measured and the methodologies available for this purpose. Studer *et al.* 2021 suggest measuring success criteria on three levels: Business Success Criteria, ML Success Criteria, and Economic Success Criteria. When applied to an AL project, besides the model performance metrics, the number of collected annotations or the quality of labels can further be a quality standard.

Finally, the conceptualization of the AL project ends with the Acceptance phase, indicating that once the defined project objectives are met, the project concludes. It simultaneously connects back to the Scoping & Project Planning Phase, highlighting its iterative nature. The inner steps, which follow counterclockwise, demonstrate various feedback mechanisms in the AL development life cycle related to the implementation, which can commence once the conceptualization is finalized.

## 3.2 Inception

When initiating an AL loop from a technical point of view, the literature often distinguishes between cold and warm starts from a technical perspective. Following this terminology, Figure 3.2 illustrates these potential entry points, each associated with a specific conceptual project phase.

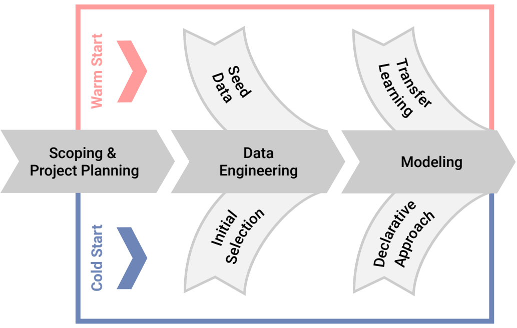


**Figure 3.2:** Potential stages of inception. After a mandatory Scoping & Project Planning phase mainly for problem understanding, tasks may vary depending on the entry point, e.g., initial sample selection, following a declarative approach, the use of seed data, or transfer learning techniques.

While a Scoping & Project Planning phase is mandatory, different methods arise depending on whether initial models or training data are already available, which can be combined. It is conceivable, for example, that an existing ML pipeline needs to be extended to include continuous training and data labeling. On the other hand, possible entry points differ in the applicable methods for project starts in situations where neither initial data labeling nor trained models are present.

### 3.2.1 Cold Start

The Cold Start of the AL project describes a scenario where the project starts from scratch, from an ML perspective, without access to information or prior experience. This means that neither a pre-trained ML model nor labeled data are available, and project participants must initially gather requirements and knowledge about the project's scope. This type of project initiation presents several challenges. Firstly, it is more difficult for project members to identify suitable algorithm types and QSs. Secondly, without prior knowledge of the data, there is significant uncertainty about relevant features and concepts, which may lead to a slower model convergence during the initial phase.

#### Initial Selection

In the literature, the Cold Start problem in an AL project is primarily examined concerning the QS for initial sample selection. The challenge framed is that without sufficient information, it becomes difficult to discern which data points are informative or representative in this early phase. However, the initial data substantially influences the efficiency and initial performance of the model. Suppose the model is initialized with data that has not been carefully preselected. It may converge to a local minimum and fail to capture the entire data distribution pertinent to the project. To address this, Barata *et al.* 2021 describe the outlier-based discriminative active learning (ODAL) approach in their paper, in which they select the most effective samples for model improvement, particularly in the initial phase of an AL project, using a novel QS based on the combination of outlier detection and discriminant analysis.

A further approach for a specialized QS is proposed by Chen *et al.* 2022, who present a solution based on contrastive learning[1]. Other methods use clustering techniques to perform the initial data selection in an unsupervised manner. For example, Brangbour *et al.* 2020 introduce a method in which cluster quality and impurity indices are additionally calculated and demonstrating that their proposed QS suits the Cold Start AL scenario.

[1] Contrastive Learning is an ML technique based on the principle of learning by comparing similar and non-similar data points.

#### Declarative Approach

As indicated in Figure 3.2, after the data-related phase follows the phase of model-related tasks. A potential entry point for the Cold Start modeling scenario lies in adopting a declarative approach. This approach encompasses methods such as AutoML, a highly probabilistic technique wherein algorithms and model architectures suitable for the underlying problem are automatically selected, extending to hyperparameter optimization and automated feature extraction (He *et al.* 2021). The high degree of automation inherent in this approach substantially expedites the model creation process, enabling individuals with limited expertise in ML to construct initial models.

In the more abstract form of declarative specification of ML code, specific facets of automation are omitted, affording developers more significant influence over the selection of ML tasks and algorithms (Molino *et al.* 2021). This approach permits more seasoned developers to make targeted adjustments and better account for specific problem domains. It establishes a bridge between automated model construction and the expertise of ML professionals, facilitating feasibility studies for more intricate challenges during this phase.

### 3.2.2 Warm Start

The second possible scenario for initializing AL is the Warm Start approach. If an ML model is already in use, introducing an AL loop can enhance the current model's performance. Compared to Cold Start, Warm Start offers several crucial advantages that become apparent when taking a closer look at the techniques that can be considered for this setup.

#### Seed Data

One approach that can be used in combination with other techniques is the use of so-called Seed Data. This refers to a previously labeled dataset on which the initial ML pipeline would be built, and the model be trained. Available data sources, even publicly available ones, can be tapped into, or a domain expert can review and label a random selection of existing data.

In the former case, it may involve sufficiently large datasets originating from another project and reused for the intended use case of the AL project. This subset of the training data is fed into an initial model with a new learning function to solve a new problem. In the latter case, randomly chosen data points are selected and annotated. The resulting labeled database serves as the basis for training the initial ML model and, as Nath *et al.* 2022 suggested, may be used to generate pseudo-labels, which are used for fine-tuning the model for the Warm Start.

### Transfer Learning

As a second entry point for warm starting an AL project, an existing, pre-trained model already possesses learned features and concepts captured in similar data or other problem contexts, leading to a quicker convergence in the learning process. Transfer learning is a well-studied method in this context, which can be combined with other methods. Pre-trained models are used as a starting point and adapted to the specific requirements of a new problem. *Pan and Yang* categorize various techniques into three categories:

- **Inductive Transfer Learning:** This setting corresponds to instance-based Transfer Learning, where it is assumed that certain parts of the data in the source domain can be reused through weighting to learn in the target domain. Instance Re-weighting and Importance Sampling are two essential techniques in this context.
- **Transductive Transfer Learning:** This category includes the feature-representation-transfer approach. An attempt is made to learn an appropriate feature representation for the target domain by transferring features from the source domain.
- **Unsupervised Transfer Learning:** This area covers parameter transfer, where parameters from the source domain are used to initialize or regulate the model in the target domain. (Pan *et al.* 2010)

For example, the combination of Transfer Learning within an AL loop has been investigated by *Kale and Liu*. The authors experimentally demonstrate that the disadvantages of the Cold Start can be overcome in an AL setup, reducing the number of required sample queries on a given dataset.(Kale *et al.* 2013)

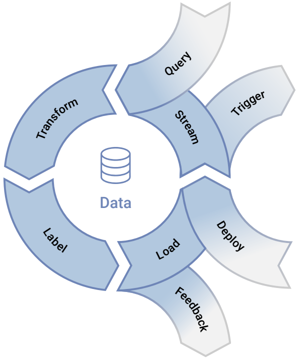
Once the initial phase is overcome, an AL project can transition into the continuous phases, which will be described in more detail in the following Section.

## 3.3 Continuous Phases

From a process perspective, the fundamental distinction between conventional AI and AL projects arises when considering the cyclic iterations and the continuity inherent in the methodology. The proposed AL development life cycle is divided into four continuous phases, each addressing tasks that iteratively occur throughout the project's duration.

The following Section details the activities and tasks targeted in each iteration and addresses the relationships between the iterative phases during implementation of an AL project.

### 3.3.1 Data Iteration



Section of a data iteration in the AL development life cycle

The phase of data iteration is a central component of the AL development life cycle and differs from conventional AI projects. Its significance lies in continuously adapting and optimizing the data foundation to ensure that the model is consistently supplied with relevant and informative data. This necessity arises from the goal of AL projects to enhance the learning process by repeatedly providing new and specifically selected data.

While AI projects may rely on static datasets, an AL project requires the continuous review and updating of data to effectively adapt the model to new information and maximize its performance. This iterative process ensures that the model is not limited to historical data but can continually learn from new, potentially relevant data and user feedback in the form of annotations.

The steps involved in data iteration simultaneously form the data pipeline, with various specifications described in the literature. When examining data engineering activities on a technical level, they can essentially be grouped into the three phases of Extract, Transform, and Load (ETL) or ELT processes (Idowu *et al.* 2021; Tamburri 2020; Ashmore *et al.* 2022; Amershi *et al.* 2019). Additionally, the entry point of a data iteration is often described as data ingestion, which consists of data collection and selection. This stage can manifest in two distinct ways in the AL development life cycle, initiating the data iteration process.

#### Query and Stream

Delving into the data ingestion phase, one can discern a bifurcation primarily characterized by pool-based and stream-based approaches. The main difference lies in how the data is processed. A stream processing engine is often used to implement stream-based data input. This component processes continuously incoming data in real-time and makes it available in short batches for further application.

In contrast to the stream-based method, querying new data in the AL development life cycle is identified as a second potential stage for entering data iteration. As presented in the main scenarios for AL in Section 2.1.3, this includes membership query synthesis and the pool-based approach. In these methods, data is not queried by the model in real-time. Instead, the model either formulates a query de novo or selects the best query set from a pool of instances.

Both approaches, stream-based and query-based, heavily depend on specific project requirements, particularly regarding data type and availability. As described in Section 2.1.3, QS plays a central role in both scenarios, aiming to identify those data points whose annotation would have the most significant impact on improving the model. These approaches can be used in isolation or in combination.

The distinction between these approaches is also reflected in *Microsoft's* TDSP development life cycle (Microsoft 2020). The differences between the methods are primarily based on how data is processed, with stream processing handling and processing data in real-time (Steidl *et al.* 2023; Kreuzberger *et al.* 2023), whereas query-based approaches focus on the strategic selection of existing data points (Settles 2009).

#### Transformation

Once new data becomes available, it is crucial to make it accessible in a consistent manner for the subsequent phases while further adhering to versioning methods. This ensures up-to-date and consistent data is processed throughout the entire AL project. Data provisioning follows a predefined sequence in which data transformation procedures are combined to optimally prepare the data for the subsequent analysis and modeling phases. During this phase, rule-based tasks are applied to the extracted raw data to improve its quality and structure. These tasks typically include noise reduction, data imputation, and the correction of inconsistencies to ensure that the data supplied to downstream ML processes is as clean and standardized as possible.

This stage includes data cleaning, which involves identifying and correcting erroneous or incomplete data. This is often achieved through algorithms and techniques that enable the interpolation or estimation of missing values, noise reduction, and detecting and treating data anomalies.

The importance of consistent data accessibility and adherence to versioning methods is emphasized in various reviews to ensure the consistency and currency of data throughout the entire process (Studer *et al.* 2021; Steidl *et al.* 2023; Kreuzberger *et al.* 2023). Data provisioning occurs in a structured sequence, with the data transformation procedures, as described e.g., by Polyzotis *et al.* 2018 and within KDD (Fayyad *et al.* 1996), also being combined to optimally prepare the data for the further stages of analysis and modeling. Rule-based tasks, such as noise reduction, data imputation, and the correction of inconsistencies, play a central role in ensuring high data quality. Ilyas *et al.* 2019 and Breck *et al.* 2019 emphasize the relevance of making the data as clean and standardized as possible for the subsequent ML processes. This aligns with the data cleaning described in the CRISP-DM model (Wirth *et al.* 2000), a central component within the data preparation phase.

### Label

In the life cycle model of an AL project, the data labeling phase is crucial and is typically defined as a separate stage in the data-related tasks. After the QS selects the data and has undergone data transformation, the new instances must be labeled by an „Oracle“. This task can be performed either by a human annotator or by the model itself as part of automated data labeling or semi-supervised learning.

The choice between manual and automated data labeling depends on various factors, including the available resources and the specific requirements of the application. Manual labeling generally provides high-quality labels but is time-consuming and costly. Automated labeling allows for faster processing but can be more prone to errors, particularly if the model is not sufficiently trained or the data is inconsistent.

Additionally, the type of application, the data type, and the problem to be solved influence the choice of labeling method. In real-time applications, automated labeling may be preferred, whereas, for safety-critical or high-precision applications, manual labeling by domain experts may be required.

The importance of this phase is highlighted in the life cycle models of Amershi *et al.* 2019 and Fischer *et al.* 2020, where it is defined as a distinct phase. The decision between manual or automated labeling and the factors based on resource requirements and the need for high-quality labels are examined by Fredriksson *et al.* 2020a. As mentioned, automated labeling is often implemented in real-time applications, while Alonso 2015 and Bernhardt *et al.* 2022 emphasize the importance of high-quality labeling, particularly in safety-critical areas. A combination of both methods is demonstrated in Desmond *et al.* 2021.

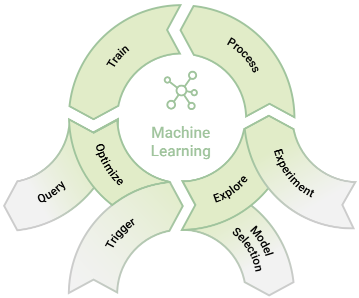
### Load

To complete the data iteration, the loading phase focuses on providing the results of data-related tasks in a standardized manner for use in other stages of the AI life cycle. During this phase, a particular consideration in AI projects is data division into training, testing, and validation datasets. Additional considerations must be taken into account as these datasets may exhibit a certain level of dynamics. New data streams and labels may be introduced, and the characteristics of the data may change over time. Therefore, it is essential to periodically review and adjust this division to ensure it remains representative of the current data distribution. These adjustments can be made, for example, through stratified sampling to maintain the same class distribution or by considering temporal aspects to capture evolving patterns.

Following the process towards the operations phase, it becomes evident that these tasks play a role in monitoring. Furthermore, artifacts resulting from a data iteration may affect different ML pipelines or should be available in additional projects that are organizationally separated but technically interconnected. Therefore, ethical and legal aspects must be considered when providing data to preserve data privacy, especially when dealing with sensitive or personal information.(Steidl *et al.* 2023)

Another task in this phase is metadata management, which documents the data and contains information about the data's origin. This contributes to understanding the context and history of the data.(Microsoft 2020)

### 3.3.2 Machine Learning Iteration



Section of an ML iteration in the AL development life cycle

Upon the availability of new data or annotations, the ML model undergoes continuous (re)training within the AL loop. This iterative process is crucial for refining the ML model's performance, ensuring that it consistently responds to the latest and most relevant information. The ML pipeline, which orchestrates this process, typically includes various sub-steps that may vary depending on the problem domain and application area, and which describe the activities of ML iteration.

These steps include data preprocessing, model training, and optimization. Each of these tasks within the AL development life cycle is designed to adapt the model to new data inputs and account for an evolving data basis. The tasks related to the ML model, including the aforementioned sub-steps, constitute the green-colored ML iteration shown in Figure 3.1.

This iterative loop is crucial for maintaining the effectiveness of the model, as it enables the model to dynamically respond to new data, annotations, or additional requirements that may arise within an AL project. The ML iteration is closely integrated with the overall AL development life cycle, with the results of each iteration feeding into subsequent rounds of data selection and annotation.

#### Trigger

In an AL system, some ML tasks are performed manually, while others are executed automatically through the continuous retraining of the ML pipeline. A central component of this automation is the various trigger mechanisms that initiate the retraining of the model.

Automated triggers can be categorized into different types. Feedback and alarm systems use feedback or alarms collected during runtime to initiate processes, such as deleting individual datasets, which can result in retraining the model. Additionally, functional components for workflow orchestration play a crucial role in automated ML processes, forming the foundation of automated ML pipeline execution. Discrete triggers, such as event-driven or time-scheduled triggers, typically drive this execution. Another example includes thresholds, where reaching certain levels triggers model updates, such as when model performance deteriorates, an effect further detailed of the „Monitoring" Section in 3.3.4. These thresholds can relate to model and data metrics, as well as throughput, latency, or GPU utilization.

This automation enables the integration of the ML pipeline into both development and operational processes. In event-driven scenarios, models are retrained when data updates occur or in response to external events, such as changes in the repository or reconfigurations of the QS. While fixed schedules may not be reactive enough and could lead to unnecessary pipeline executions, they still offer advantages in optimizing the frequency of retraining, resource allocation, and the order of job executions, particularly when edge and cloud resources are involved. This dynamic not only underscores agility but also promotes proactive readiness. Fixed time triggers (e.g., hourly, daily, weekly) ensure that models are regularly retrained. This consistent approach ensures that the models remain adapted to the current data landscape, maintaining their relevance and performance.

The importance of automated trigger mechanisms for the continuous execution of the ML pipeline is highlighted by Steidl *et al.* 2023, who identify three main types: feedback and alarm systems, orchestration services, and schedules, as well as repositories. Kreuzberger *et al.* 2023 emphasize in their end-to-end MLOps architecture the significance of functional components for workflow orchestration that support an automated ML pipeline. In this context, they highlight discrete triggers, such as event-driven or time-driven triggers, and thresholds that trigger model updates, which are central to making the ML pipeline dynamic and efficient.

### Experimentation

The interface to the development iteration refers to the experimentation phase, which further constitutes a substantial phase within the ongoing process in AL projects. Experimental tasks encompass activities such as evaluating novel model architectures, quantifying model performance with parameter configuration adjustments, or exploring innovative approaches to data preprocessing. The iterative nature of this process allows for the continuous incorporation of new ideas and improvements into model development.

Activities such as feasibility studies and concept proofs may be included within this phase, stemming from an ML iteration, as proposed by Haakman *et al.* 2021. This means that AL teams have the opportunity to test new concepts and approaches in an experimental framework, verifying their applicability and effectiveness before transitioning into actual model production (Baylor *et al.* 2017).

Moreover, the subtle shift from the developmental iteration towards the experimental corridor of ML iteration underscores a quintessential tenet for AL projects, as elucidated by Mattos *et al.* 2017: the orchestration of automated continuous experiments. This paradigm facilitates methodical hypothesis validation within the ML context and underscores that AL ventures transcend episodic experimental bouts. As a result, AL projects are not limited to one-time experiments but can conduct continuous testing and experimentation to monitor and improve model performance over time (Hummer *et al.* 2019). This contributes to creating adaptive learning systems that adapt to changing conditions and requirements, thus maintaining long-term effectiveness.

### Exploration

Irrespective of an ML pipeline's application-specific implementations, one stage involves a systematic data analysis. When the ML iteration commences in an experimental milieu, manual execution becomes typically required, similar to how it is described for the conceptual Data Engineering phase (cf. 3.1.2). Here, experts meticulously dissect the data, leveraging visual analytics to decipher inherent patterns and intricate relationships that hold promise for algorithmic training. A case in point is the deployment of attribute recognition methodologies. These are systematically harnessed to probe the data, mining for salient attributes that can be seamlessly integrated into the model's fabric.

Some tasks in this phase are amenable to automation, especially when navigating familiar data relationships. Although the tasks are data-driven, the focus here shifts towards a model-centric orientation. In this context, the data corpus undergoes statistical evaluations, enabling a profound understanding of its inherent attributes and distributions. Complementing this, anomaly detection techniques stand poised to identify and mitigate potential data aberrations, ensuring they don't inadvertently compromise the integrity of the resulting model.(Polyzotis *et al.* 2018; Caveness *et al.* 2020; Breck *et al.* 2019)

### Processing

The subsequent processing steps in the proposed AL development life cycle are categorized under „Process". The intricacies of data preprocessing for the ML model are closely tied to domain-specific considerations and the overarching application context. The relevance and effectiveness of specific techniques vary depending on the data type and the model's architectural specifics. A strategic combination of various methods is often orchestrated to prepare the data for model training optimally.

This preparatory phase includes methods such as data weighting and resampling. In data weighting, different levels of importance are assigned to certain data instances to increase the significance of data subgroups selectively deemed critical for model performance. Resampling, conversely, involves a tactical reconfiguration of the data distribution to reduce the dominance of overrepresented data points.

Another crucial component of this phase is feature engineering, where new, potentially more informative features are developed from the existing data pool. These procedures range from mathematical transformations and aggregations to targeted feature selection and extraction areas. Before these refined data are fed into an ML model, normalization procedures ensure consistency in value ranges, while standardization techniques, such as scaling, provide a uniform data structure. All these preprocessing tasks must align with the application in the production environment to prevent training-serving skew.

Another central aspect of preprocessing is data augmentation, a critical strategy in modern ML practice. Traditional augmentation techniques include data shifts, such as rotations, scaling, or mirroring of the original data, to increase data diversity. Other augmentation methods introduce controlled perturbations, such as injecting noise or overlaying artifacts, to enhance the model's robustness and generalization capabilities. Another augmentation method is data synthesis, where new data instances are created from the existing corpus, often using additional ML models for this generative process.

These preprocessing steps' importance and adaptation to the specific application context are comprehensively integrated into established process models. Studer *et al.* 2021 emphasizes the close connection between data preprocessing and domain-specific requirements, where the effectiveness of techniques varies depending on the data type and model architecture. Idowu *et al.* 2021, Amershi *et al.* 2019, and Arnold *et al.* 2020 support the inclusion of feature engineering during this phase to enhance model performance by creating new features. Steidl *et al.* 2023 highlights that normalization and standardization procedures are crucial to ensure consistency between training and production environments and to avoid training-serving skew. Finally, the significance of data augmentation in improving model robustness and generalization through various techniques, such as data shifts and data synthesis, is detailed by Studer *et al.* 2021.

### Training

The subsequent phase shifts the focus from the data towards the model. Training represents the pivotal step within the ML iteration, during which the adjustment of the ML model is carried out according to its objective function, mirroring the defined problem-solving approach. In this process, pre-processed data is fed into the model to identify patterns. The outcome is an algorithm that is subsequently deployed as a prediction service in an AI application and is iteratively retrained within the context of an AL project. Thus, the training phase aims to refine the model so that the resulting algorithm can make as accurate predictions as possible on unknown data points.

Similar to pre-processing methods, choosing appropriate modeling techniques and algorithm types is largely contingent upon the nature of the ML problem, the set objectives, the data, and the project constraints. The overarching objective, often serving as a benchmark for model performance, acts as a proxy. Another essential facet within this training process is the optimizer. It dictates the strategy of the learning process and determines how the model parameters are calibrated to meet the defined objectives. Moreover, regularization is an integral component of the training process. It can be incorporated into the objective function, the optimizer, and the model itself, playing a crucial role in minimizing overfitting. Through it, more distinct and stable solution paths in the learning process can be identified.(Amershi *et al.* 2019; Studer *et al.* 2021)

### Optimization

In the subsequent phase, optimization and regularization techniques are intensified. Model-oriented optimization, which represents an extension or additional iteration in ML, typically involves activities focused on the iterative repetition of the training process and improving model performance. An essential task in this phase is hyperparameter tuning, where a range of external configuration variables, such as the learning rate, are systematically tested, as they significantly influence an algorithm's learning process. In this context, model performance evaluation is usually carried out using a validation dataset, which is either provided by a previous method or generated in real-time during this phase, for example, through techniques like cross-validation.

If the design of the ML pipeline includes the training of multiple models, as is the case with ensemble methods, methods for aggregating these models are implemented during this phase. Another typical activity during the optimization phase is model compression, where compression or pruning techniques are applied to create a more compact model. These procedures aim to prepare the often significantly large trained models for subsequent use, either to enable faster inference times through accelerated predictions or to reduce the required storage space for integration into an edge device. Furthermore, in the AL context, this optimization can include parameterized QS, which is optimized during a labeling simulation.

Other established life cycle models address the importance of model-oriented optimization and its various activities. For example, Zaharia *et al.* 2018 describe the iterative repetition of the training process and the optimization of model performance as central tasks. Ashmore *et al.* 2022 emphasize the systematic review of hyperparameters that govern the learning process of the algorithm. Furthermore, the necessity of model compression during the optimization phase is discussed by Studer *et al.* 2021 and Karlaš *et al.* 2020, to prepare models for more efficient use, whether through faster inference or reduced storage requirements.

### Model Selection

The model selection phase in the AL process serves as the connection between the ML iteration and the development iteration. After optimization, once a suitable version of the fully trained model has been identified, the focus shifts to deciding which model is best suited for the specific requirements of the project in the subsequent phases. In this context, it is crucial to implement a model versioning system that not only tracks the different versions of the model artifacts but also their dependencies. This allows for easy reproduction or rollback to previous versions as models rapidly evolve.

During model selection, certain criteria, such as complexity, interpretability, and scalability, are manually evaluated and incorporated into decision-making during the conceptual modeling phase. Other important criteria, such as compatibility and integration, arise from the requirements of the software environment or the integration into existing data processing pipelines. On the other hand, performance factors like predictive accuracy or robustness can be managed through automated processes.

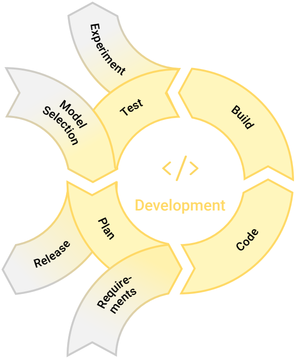
This phase is particularly significant in the context of AL, as it involves the continuous adaptation of the model through the ongoing expansion of the training data. Clearly, the data selected by the QS and the chosen model influence each other. A successful model selection strategy thus ensures that the model used in the next iteration of the AL process is both powerful and robust, accurately reflecting the current data situation.

The literature on existing life cycle models extensively discusses the importance of model versioning and the criteria for model selection. Steidl *et al.* 2023 and Zaharia *et al.* 2018 emphasize the technical necessity of a model versioning system that facilitates tracking and restoring previous model versions. Inspired by Wirth *et al.* 2000 , Studer *et al.* 2021 focuses on the model selection stage by choosing an appropriate model architecture based on various performance factors such as accuracy and robustness.

### 3.3.3 Development Iteration

Concepts of iterative development cycles originate from the practices of DevOps and encompass methodologies for the continuous improvement and automation of conventional software projects. In traditional software development, DevOps pipelines already consist of an array of methodologies employed by development teams to design, test, and deploy software products more efficiently and promptly.

The AL development life cycle adapts the concepts of continuous integration and deployment of the source code and ML-related artifacts. These include the trained model, which has emerged from the ML iteration and is made available for the subsequent development iteration according to the model selection phase.



Section of the development iteration in the AL development life cycle

#### Requirements

As in traditional software development projects, it is crucial in AL projects to ensure that the developed system, the model to be trained, and the generated data align with the project goals and the actual needs of the users. During an ongoing requirements analysis, feedback is continuously consolidated from an extended circle of stakeholders. This circle can range from subject matter experts to end users to ensure that diverse perspectives and expertise are incorporated into the development process. This phase aims to quickly identify emerging or changing requirements and integrate them into project planning.

This aspect is particularly important because the dynamics of new data and models further influence the development and refinement of models and other software components. However, the foundation for all requirements is the use case, including its users and their needs. To effectively capture these, methods such as user interviews and in-depth user research can be conducted to identify requirements related to user interaction, user interfaces, and feedback mechanisms.

The engineering team must also address data-related requirements, such as availability, access, quality, volume, and data protection. The same applies to requirements related to annotation, which directly influence the design of annotation tools so that users can efficiently label data. These identified requirements can impact the entire spectrum of the AL project and span its entire lifecycle. While some requirements may concern the ML model, others might affect data and QSs. Still, other requirements might have less influence on ML-related areas and focus primarily on the user experience. Therefore, careful planning is essential before diving into the development iteration. The importance of ongoing requirements analysis and the inclusion of a broad stakeholder circle is emphasized in the literature.

Rahman *et al.* 2019 highlights the need to incorporate different perspectives into the development process, while Steidl *et al.* 2023 describes the dynamics of new data and models and their impact on development. The collection and categorization of requirements, particularly for ML applications, is comprehensively discussed by Studer *et al.* 2021, who summarize model-related requirements in ML projects arising from the operation of an application in the target environment into dimensions such as performance, robustness, scalability, explainability, and resource requirements. Another categorization is presented by Habibullah *et al.* 2023, who identify 35 groups of non-functional requirements for ML applications and classify them according to product operation, product revision, and product transition.

### Plan

In the planning phase of the AL development life cycle, a further initial step in the development iteration emerges. This step in the process primarily addresses the requirements that can be derived from the feedback gathered during the operational phase. In addition to this predictable flow, the high volatility introduced into the AL project by the constantly changing data and model landscape can lead to requirements that necessitate spontaneous responses and adjustments by engineering teams.

Typically, the transition to the continuous phase occurs systematically, with task-tracking tools frequently being utilized. These instruments serve the purpose of organizing and monitoring assigned tasks. In this context, a established DevOps and MLOps principle becomes a cornerstone: The collaboration of project members. Teams define objectives based on the specific areas and structure the implementation into tasks by setting priorities, calculating available resources, and considering time constraints. In this phase, it is paramount that interdisciplinary teams collaborate closely to foster a consolidated understanding of the multifactorial requirements directed at the ML model and other system components and how these might integrate into pre-existing parts.

### Code

The next step in the development iteration focuses on the actual implementation. In this phase, the previously defined specifications are transformed into functional software code. Particularly in the context of ML, this process encompasses the conventional programming code as well as the code of ML-specific algorithms, which represent the probabilistic aspect of an AI system and are organized within the ML pipeline.

During this phase, it is paramount to ensure close collaboration among the often diverse development teams. Regardless of which component of the AI project is being developed, whether it pertains primarily to data engineering functions or the ML pipeline, the involved team members use integrated development environments (IDEs), code editors, and other appropriate technologies to convert the planned modifications into executable source code. In coding for ML projects, the use of codespaces has become established, with environments being created, for instance, through the use of a development container. These development containers, also known as „Dev-Containers“, are Docker containers that are specifically configured to provide a fully functional development environment (GitHub 2023).

### Build

As evident in Figure 3.1, the next step in the yellow colored development iteration is a new version's build. In this stage, the developed code is turned into an executable form, an essential part of continuous integration, which according to Karlaš *et al.* 2020 is already the „de-facto standard for building industrial-strength software“.

In software development, it is expected to utilize a variety of third-party libraries and frameworks. Dependency management is used to resolve the versions of the packages used and integrate them into the project. During this phase, the model is orchestrated with the rest of an application's logic at the system level. Part of this process is called packaging, where the code and model logic are bundled into build artifacts that are used for further steps, particularly for later deployment. Thus, the essential task is containerizing the model and its dependencies to prevent compatibility issues (Lenarduzzi *et al.* 2021).

### Test

As in traditional software development, a successful build is followed by a phase in which manual and automated tests are conducted. These tests are designed to ensure that the artifacts created during the development iteration meet functional and quality requirements. The testing phase bridges the gap between the development iteration and the subsequent phases, and it precedes the release phase, highlighting that continuous testing is an essential step in the AL development life cycle.

In data-oriented projects with an ML pipeline, automated tests, including unit tests, integration tests, and regression tests, extend beyond the application code to encompass data dependencies and the selected model. Various ML-specific testing properties must be considered, such as correctness, model relevance, robustness, security, privacy, efficiency, fairness, and interpretability, as proposed by Zhang *et al.* 2019.

The importance of the testing phase and the necessity of continuous testing are emphasized in the literature. Gmeiner *et al.* 2015 highlights the role of manual and automated testing following a successful build, while Sculley *et al.* 2015, Tao *et al.* 2019, and Lenarduzzi *et al.* 2021 emphasize the importance of ML-specific tests that go beyond the application code and cover data and model dependencies.

While Breck *et al.* 2017a proposes a comprehensive approach to ML testing at the project level, covering the entire development life cycle, Steidl *et al.* 2023 discusses the challenges of automating tests in ML pipelines. The available testing strategies show that achieving full automation makes it difficult to ensure efficient use throughout the pipeline. For example, acceptance tests often require user involvement, which complicates complete automation.

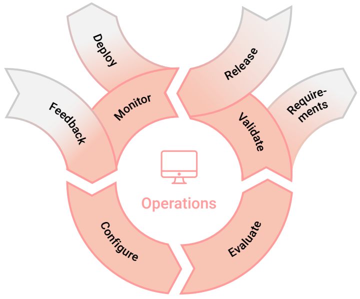
**Release**

Following the successful completion of tests for the selected model and the implemented code, the process advances to the release phase. The pertinent software and model versions are securely and efficiently transitioned into the operational phase during this phase. At this juncture, the development team typically produces all essential documentation required for operation. This lays the foundation for proficient change management, fostering development iteration in the AL life cycle through a culture of continuous improvement.

A crucial component within the continuous integration and delivery paradigm that impacts this phase is formulating a rollback plan encompassing all necessary technical aspects. This ensures that the AL life cycle remains agile, instilling confidence in executing future development iterations. Furthermore, the team responsible for the operational phase can respond should issues arise, reverting to a prior version introduced in an earlier release phase.

### 3.3.4 Operations Phase

In addition to the iteration cycles, representing data engineering, the ML pipeline, and development, all operational tasks are highlighted in red in Figure 3.1. These tasks are carried out continuously in a production environment from the time of release and throughout their deployment. In a sense, this phase bridges the gap from the development iteration back to the data iteration, emphasizing the data-centric focus of the AL life cycle. The operational tasks emerge with significant importance, as they contribute to the improvement of quality and the enhancement of efficiency and further represent the essential aspect of the feedback cycle for an AL application.



Section of the operations phase in the AL development life cycle

**Feedback**

When models of AI systems undergo updates over time, it often induces alterations in the system's inherent behavior, a phenomenon articulated by Sculley *et al.* 2015. In their discourse, they shed light on the complexities birthed by such changes, leading to what they term as a form of „analysis debt“. This characterizes a predicament where anticipating a model's behavior before its official deployment becomes an intricate endeavor. Echoing this sentiment, Kreuzberger *et al.* 2023 underscore the imperative of integrating multiple feedback loops within an AI system. Their rationale pivots on the notion that feedback loops act as conduits, channeling insights from quality evaluations back into the iterative engineering processes.

Examined through the lens of AL, the paramountcy of continuous feedback throughout the life cycle emerges distinctly, cementing its role as the central nexus between the oracle and the learner. These direct, explicit feedback loops, especially in an operational setting, encapsulate responses from stakeholders interfacing with the deployed system. This entails responses and actions captured as data annotations for the requested samples, which are harnessed for the model's subsequent training phases. This facilitates the evaluation and validation of the learning progress, offering indicators, for instance, of data where the model showcases a high prediction performance or segments of the problem sphere necessitating further conceptual acquisition. This feedback equips the operations team, allowing them to enact initial responsive measures, such as system reconfigurations or, for example, recalibrations of the QS.

While direct feedback loops may be resource-intensive during analysis, they invariably pose a persistent challenge in the AL life cycle. A more intricate scenario emerges from latent feedback loops arising from indirect system influences. The ensuing implicit feedback manifests, e.g., in behavioral shifts, evident when users, in response to a new model version, either intensify or diminish the frequency of anticipated actions (Sculley *et al.* 2015).

Across both scenarios, the imperative remains the meticulous collation of information regarding the model's performance juxtaposed with user behavioral patterns during operational deployment, all of which inform a robust monitoring paradigm.

**Monitoring**

With the integration of continuous monitoring, another MLOps principle is implemented into the AL development life cycle. During this phase, there's a dual focus: firstly, on overseeing the current state of the learning model, and secondly, on ensuring the efficiency of the overall learning process. Continuous monitoring offers the ability to identify vulnerabilities and inconsistencies during training, allowing for timely interventions and corrections.

This approach implies consistent oversight of data quality, including new annotations, as well as tracking the learning progression using model performance indicators. These indicators evolve over time, such as accuracy, loss, and other relevant metrics. A sudden decline or stagnation in these indicators can point towards challenges in the training process that might affect the learning progress. Due to possible shifts in the data basis, changes in user behavior, or other environmental factors, the statistical properties of target variables, which the model aims to predict, can change over time. This phenomenon is generally understood as concept drift (Gama *et al.* 2014). In this context, specialized detection techniques are employed with the objective of preserving the predictive performance of the model as it evolves.

An exhaustive approach to continuous monitoring doesn't solely revolve around data and the model. Furthermore, it encompasses other pivotal components of an AI system (Kreuzberger *et al.* 2023). Regular assessments of the code and infrastructure resources, for instance, are crucial. It ensures that they continue to meet the demands of the entire AL process and, if necessary, are modified to cater to new requirements.

### Configuration

As evident in the AL development life cycle, the stages Feedback and Monitor converge into the Configuration phase. To achieve optimal performance from a model, insights from these stages must be assimilated and meticulously refined. Such refinements often involve adjusting parameters to calibrate the model's architecture to new datasets and labels.

Sculley *et al.* 2015 emphasize the significance of configuration in AI systems. They note that large-scale systems possess an array of configurable options, a fact apparent from the previously examined data-, ML-, and development- iterations. These range from data acquisition, data transformation, feature selection, potential pre-processing or post-processing methodologies and, model training settings.

Consequently, the chosen configuration in an AI system influences the model, learning process, and the QS. The operations team should be equipped to respond flexibly to shifting dynamics. Switching to an implemented QS should be facilitated by a reconfiguration, bypassing the need for a development iteration. This flexibility allows the system to adapt to available resources and factors influencing this decision, informed by outputs from the Monitoring- and Feedback-stage.

However, the versatility of configuration management poses its set of challenges. Real-time modifications necessitate a reevaluation of outcomes. A continuous assessment mechanism is indispensable to ensure that all configuration adjustments consistently align with the overarching goal of optimal model performance.

### Evaluation

In addition to assessing the current configuration, the continuous evaluation of the entire system's performance is essential due to the ongoing changes in training data and continuous retraining (Kreuzberger *et al.* 2023). At this juncture, the effectiveness of the trained model in making predictions on a test dataset is examined. Typically, a subset of data that is always disjoint from the training and validation datasets is utilized to ensure an unbiased evaluation akin to a „blind test" (Studer *et al.* 2021). Studer *et al.* 2021 further advise that this test dataset be carefully selected and curated, preferably by a team of domain experts who can verify accuracy and representativeness for real-world cases. Specific challenges in implementing the AL loop might arise, as care must be taken throughout the project's duration to ensure that the dedicated test data were not used as training data in an earlier phase.

Depending on the problem statement and model type, performance metrics, such as accuracy, $F_\beta$-score, precision, recall, and the receiver operating characteristic (ROC) curve are measured on this dataset. Each of these statistical indicators captures distinct facets of model quality by quantifying prediction accuracy. Evaluation criteria are established based on the ML objectives of the project, such as achieving a specific prediction accuracy for a certain class within the dataset. Beyond these qualitative scores, other model attributes can be assessed, equally pertinent in determining the model's performance, such as prediction speed or costs related to required training duration or storage capacity (Zaharia *et al.* 2018). Regardless of which aspect of the model is being measured, it is inherent that some values can be computed metrically while a set of values needs to be estimated. For this purpose, dummy estimators, for instance, are employed, simulating a model that generates random predictions as a baseline.

The QS is the second ML-related performance category subjected to continuous evaluation in the AL context. This phase investigates whether the choice and configuration of the selection strategy align with the desired objectives, for example, if a forced class selection effectively expands the data space for the model in this task and the desired effects on the model, e.g., a higher certainty value, become apparent. Thus, this phase provides insights into whether training and additional labeling of training data should continue or if the model has already achieved an acceptable performance level. Suppose performance gains between iterations are only marginal. In that case, it might indicate that the model benefits little from further annotation rounds, necessitating further design decisions - a potential feedback loop linking the operational phase with the development iteration through the requirements stage.

### Validation

Evaluating the model's performance and the QS provides an initial insight into the performance level achievable with the current conditions. However, more than simply computing metrics is required to ensure a model makes accurate predictions. Hence, during the operational phase, additional efforts are made to determine whether the system meets practical requirements.

Continuous validation in an AL development life cycle ensures that the model and data align with the expected operational behavior in real-world conditions (Caveness *et al.* 2020). As mentioned in the Model Selection- (cf. 3.3.2) and Test-stage (cf. 3.3.3), the criteria for interpretability and robustness play a pivotal role in this context. For example, utilizing model explanation techniques can aid in understanding „how“ and „why“ the model makes specific predictions and analyze the model's functioning in-depth.

Besides model validation in AL projects, another crucial aspect is the continuous validation of the provided annotations. One method in this regard is, for instance, determining inter-annotator agreement. Here, a sample of annotated data is taken and compared with additional annotation sets from other oracles to validate the quality and accuracy of the assigned labels, ensuring high consistency in data annotation.

### Deployment

The deployment phase, which connects the operational phase with the data iteration in the development life cycle, embodies the characteristics of the AL methodology. This phase focuses on activities to integrate artifacts such as traditional software components and model versions into a production environment where they are available for interaction with the Oracle.

Best practices in continuous deployment include safety techniques such as A/B testing and canary releases (Zinkevich 2019). These techniques minimize risks and allow new model versions to be gradually introduced and tested before full implementation. In the context of the release phase, automated processes for rollback decisions can be employed to reduce the risk when introducing a new model version.

For the deployment phase of an ML model, Studer *et al.* 2021 identify relevant activities that go beyond risk minimization and the definition of a release strategy. These include defining the inference hardware, addressing infrastructure concerns, with significant differences depending on whether the target platform involves scalable cloud-based services or embedded systems.

Steidl *et al.* 2023 identify four essential criteria a model must meet before deployment. First, all steps in the development pipeline, including quality assurance, must be successfully completed. Second, the optimal model must be identified, which can be particularly challenging when multiple model versions need to be fairly compared. Third, custom deployment criteria, such as improving accuracy or achieving a specific benchmark, must be met. Fourth, human involvement may be necessary, such as in manual testing, particularly with specific participants.

The importance of these criteria and continuous deployment techniques is extensively covered in the literature. Baylor *et al.* 2017 and Olston *et al.* 2017 discuss best practices for risk minimization and automation in the deployment phase, while Studer *et al.* 2021 and Steidl *et al.* 2023 mention the specific requirements for inference hardware and the need to meet custom criteria.

## 3.4 Roles

Incorporating AL into a development project, especially when aiming for a high degree of maturity, requires the integration of a variety of roles, encompassing interdisciplinary groups of both technical and non-technical participants. This Section aims to provide a general overview for assigning the roles used in the subsequent Chapters.

### 3.4.1 Non-technical Roles

**Project Managers.** Falling under the umbrella of Business Stakeholders, they oversee the entire life cycle of the AL project. Project Managers coordinate the various roles, manage resources, and ensure project milestones are met in time and budget (Lwakatare *et al.* 2020b). They play a key role in risk management and aligning the project with its objectives.

**Ethicists.** They accompany the development process and ensure that all processes within an AL project comply with ethical standards and legal regulations through methods such as „Embedded Ethics" (McLennan *et al.* 2020). This includes monitoring the fair and unbiased use of models and conducting ethical reviews of data labeling.

**Annotators.** In AL projects where human oracles support the labeling process, these individuals may bring various qualifications or be trained to annotate specific data types. For labeling tasks that do not require specialized knowledge in a field, a crowdsourcing approach is often adopted in large-scale projects (Nguyen *et al.* 2015; Zhao *et al.* 2011). Annotators play a pivotal role by supplying the model with requested annotations and providing feedback throughout the iterative learning process (Budd *et al.* 2021; Fredriksson *et al.* 2020b).

In an AL project, human oracles, who may bring various qualifications or be trained for labeling certain data, are deployed (Budd *et al.* 2021; Fredriksson *et al.* 2020b). They play a crucial role by providing the requested labels and feedback for the model.

**Domain Experts.** Subject Matter Experts are individuals with in-depth knowledge essential for comprehensively understanding the problem domain and seeking an appropriate algorithmic solution. They often act as a bridge between non-technical and technical roles, for example, by refining labeling guidelines with Data Scientists to ensure high-quality annotations and consistency of labeling (Budd *et al.* 2021).

### 3.4.2 Technical Roles

**Data Scientists.** Data Scientists are responsible for designing algorithms and modeling. Their role typically encompasses all tasks from the ML iteration, including feature engineering as pre-processing, implementing the logic for model training, optimizing models, and implementing QSs.

They work closely with other team members to ensure that the AL process is aligned with project goals.(Lwakatare *et al.* 2020b; Kreuzberger *et al.* 2023; Makinen *et al.* 2021)

**Data Engineers.** Their realm lies mainly in data iteration steps, such as procuring, organizing, and managing the datasets used for the AL project (Kreuzberger *et al.* 2023). They ensure that the data is clean, well-annotated, and organized to be accessible for the AL process, and therefore primarily trained for tasks in the field of DataOps (Tamburri 2020).

**Data Curators.** Data Curators manage and enhance data collections to ensure they are organized and accessible for the projects use case. Their responsibilities include assessing and improving data quality, correcting, enriching sources, and managing metadata. Collaborating closely with Data Engineers and Data Scientists, they support data integration for model training and -analysis. They ensure data sets are well-documented and consistently formatted, supporting data lineage traceability and compliance with governance standards.(Tammaro *et al.* 2019)

**Solution Architects.** This group includes roles concerned with the design and technologies of the system. Software Engineers ensure adherence to best practices and coding guidelines in the implementation. DevOps Engineers are instrumental in automation, for example, for CI/CD (Lwakatare *et al.* 2020a). Quality Assurance or Test Engineers ensure that implemented components function as intended, creating tests to potentially identify system errors (Lenarduzzi *et al.* 2021).

**AI/ML Engineers.** They are responsible for integrating ML models into a production environment. As a cross-domain role in MLOps, they take components, such as the model from Data Scientists, and ensure seamless integration into the scalable production system by managing the operation of the infrastructure, for example, through cloud resources, automated execution of pipelines, provisioning, and monitoring (Kreuzberger *et al.* 2023; Lwakatare *et al.* 2020b; Makinen *et al.* 2021).

Each role contributes unique skills and knowledge, and the synergy between these roles lays the foundation for the successful development and deployment of AL systems.

A proposal for a development methodology to enable effective collaboration between the different technical roles is presented in the following Chapter 4.

## 3.5 Summary

This chapter introduces the development life cycle for AL and explores how modern concepts from DataOps, DevOps, and MLOps can be integrated into AL projects. In light of traditional process models and AI/ML life cycles frameworks, this part of the dissertation proposes an iterative approach that facilitates the transition to an agile development process.

For this purpose, the phases and individual steps as well as possible starting points of an AL project were defined, and the characteristic tasks within these steps were structured. By clarifying their interrelationships, the conceptual phases and the continuous iterations in the areas of data, ML, development, and operations were aligned. Furthermore, relevant technical and non-technical roles were identified.

# Engineering Methodology 4

**4.1 Developing in Data-Centric Projects . . . . . 86**
**4.2 Basic Concepts . . . . . 86**
4.2.1 Data Principle . . . . . . 87
4.2.2 Code Principle . . . . . . 89
4.2.3 Automation Principle . . 90
**4.3 Branching Workflow . . 92**
4.3.1 Main Branch . . . . . . . 92
4.3.2 Feature Branches . . . . 93
4.3.3 Experiment Branches . . 94
4.3.4 Release Branch . . . . . . 94
4.3.5 Production Environment 95
**4.4 Git Workflow for Active Learning . . . . . . . . . 96**
**4.5 Evaluation . . . . . . . . 99**
4.5.1 Literature Based Approach . . . . . . . . . . . 99
4.5.2 Expert Interviews . . . . 110
4.5.3 Metadata Analysis . . . 116
**4.6 Summary . . . . . . . . . 120**

In the implementation of AI systems, the technical roles delineated in Section 3.4.2 are confronted with a plethora of emerging challenges, particularly evident as projects progress from developing feasibility studies to large-scale, mature projects.

These challenges are multifaceted, extending from the necessity of harmonizing collaboration among diverse roles — each with its distinct methodological approach — to the details of managing data versioning, ensuring the traceability of performance metrics, and securing the integrity of resulting artifacts such as the trained model (Arpteg *et al.* 2018; Amershi *et al.* 2019; Lwakatare *et al.* 2020a; Fischer *et al.* 2020; Tamburri 2020; Diaz-de-Arcaya *et al.* 2023).

The preceding Chapter 3 meticulously maps out the structured life cycle dedicated to the development of AL projects, articulating the array of tasks the team navigates through the distinct phases and iterations of the project. Advancing beyond this process model, the following Chapter 4 delves into strategizing how these interdisciplinary teams can synchronize their efforts to implement AI systems effectively and efficiently. This exploration is rooted in a commitment to address the existing challenges by instructing the engineering team with precise, actionable guidance distilled from the flexible and agile nature of the AL development life cycle.

To this end, individual concepts of a development methodology are proposed and presented as a synthesis of theory and practical application, serving as an interface between the working methods of the different technical disciplines.

The methodology was first introduced in the publication Stieler *et al.* 2023a, with a concentrated lens on AL projects, evaluating its practicality through an examination of best practices adherence, augmented by insightful findings from interviews with experts from the industry. The concepts presented were abstracted and described in more detail in Stieler *et al.* 2024 for data-centric AI projects. Furthermore, the evaluation of the methodology was extended on the basis of a metadata analysis, which is based on the technical implementation of the respective AL projects of the use cases described in Chapter 6.

## 4.1 Developing in Data-Centric Projects

Analogous to the development of concepts such as DevOps, in traditional software development, an ecosystem of tools and methods has emerged, which further includes best practices and development strategies like Git-Flow (Driessen 2010) and trunk-based development (Zettler 2023). These methods enable teams to collaborate on different parts of a project and, through the use of version control systems, develop high-quality software by tracking project progress and integrating or, if necessary, reverting changes made by multiple contributors (Chacon *et al.* 2014).

However, these solutions cannot be directly applied to AI projects, as AI and traditional software projects fundamentally differ in one aspect: in traditional software projects, the source code is sufficient to create the artifacts (e.g., an executable program). AI projects, whose artifacts include a trained ML model, have two types of inputs: code and data (Sculley *et al.* 2015). Since data is usually more volatile than code, ML artifacts need to be recreated more frequently. In this context, emerging concepts are currently establishing themselves in the form of best practice formulations for AI projects and the formalization of MLOps principles (Serban *et al.* 2020; Kreuzberger *et al.* 2023).

The agile development methodology outlined in this Section offers a set of guidelines designed for teams of interdisciplinary technical roles, like software engineers and data scientists, to systematically organize their work. This methodology emphasizes a data-driven development ethos, acknowledging the central role of data in AI projects and the unique challenges it presents. It aims to strengthen collaboration and enable traceability of artifacts in terms of code and data for the entire engineering team to help meet the requirements of trustworthy AI development.

A central idea of the proposed development methodology is the integration of runners for CI, CD, and CT[1], which each behave differently depending on the task for which they are intended and are therefore crucial for the agility and effectiveness of the AI system development process. These runners automate key development tasks, ensuring that new code commits trigger a cascade of actions, from automated testing to model training and deployment, thereby enabling rapid iteration and feedback.

In Section 4.3, we elaborate on a branch-based workflow in which we introduce data- and code focused levels as well as new types of branches. To realize this, we first present the necessary principles related to data, code, and the runners.

## 4.2 Basic Concepts

In the current landscape of AI system development, a multitude of technologies and toolchains are emerging that aim to extend the capabilities of developers and data scientists (Giray 2021; Ruf *et al.* 2021). As these technologies advance, it becomes essential to formulate methods independent of any specific tools, ensuring broad applicability and integration compatibility across various platforms and environments.

[1] Following Steidl *et al.* 2023, the acronym CD4ML is frequently used in this context, which describes the technical implementation of MLOps concepts by applying CD principles to the AI lifecycle management.

The concepts and methods outlined below are designed not to be tied to particular tools or technologies but to provide a universal approach that can be applied to a branch-based development process[2]. Even though established tools, whether open-source or proprietary, can further significantly influence productivity and collaboration, the proposed methodology aims to streamline the workflow in data-centric AI projects. It does so by allowing the use of any technology stack, thereby avoiding the dependence on specific tools. This universality ensures that the branch-based development approach, coupled with an agnostic attitude towards the burgeoning variety of tools, creates a workflow that accommodates the diversity of the current technological ecosystem.

To implement the foundational concepts, such as the use of runners for CI, CD, and CT to enable automated iterations as proposed in (Karamitsos *et al.* 2020), the following Sections will present fundamental principles related to data and code management. In addition, we provide a brief overview of the necessary infrastructure.

### 4.2.1 Data Principle

As described in Section 3.1, after the initial conceptualize phases, from a technical process perspective, AI projects typically start with collecting and preprocessing raw data. Like the program code, the data exhibits a persistent dynamism across the project's duration. These changes can result from adding new data results and new project requirements, which in turn require the collection and processing of additional data. In practical terms, new raw data may need to be imported or new data labels introduced in a subsequent iteration phase.

Here, it becomes clear that the artifacts generated by an AI project do not solely depend on the underlying code but are significantly influenced by the data used and processed. This necessitates efficient and traceable data versioning (Kreuzberger *et al.* 2023; Karlaš *et al.* 2020; Amershi *et al.* 2019). Although it is possible to do this manually – for example, by assigning timestamps to the different versions of the dataset – a dedicated tool for data versioning is often recommended in practice to ensure optimal transparency and reproducibility, in particular when certification and qualification of AI systems are addressed.

Typically, tools like Data Version Control (DVC) store data as unique snapshots or as raw data in external storage systems (e.g., an object or file storage) and link them to the project repository in the using a metadata file (Iterative 2020). Other technologies would be applied when the data volume becomes too large for the aforementioned method, implementing by versioning the data changes made. Yet another approach conducts location-dependent data storage, for instance, when using edge devices. In this context, it becomes clear that it is important to version the dependencies of the processing steps and the extracted features (Lwakatare *et al.* 2020b; Kreuzberger *et al.* 2023). These are, for some use-cases, stored in Feature Stores to maintain consistency and avoid redundancies.(Steidl *et al.* 2023)

Developing an ML pipeline with complete, often massive datasets can make implementation inefficient for various reasons. Factors such as data selection and extended computation time can significantly slow down the

[2] Tools used to apply the development methodology require the support of a branch-based structure for code and data.

engineer's workflow. For this reason, we recommend creating a so-called development dataset. This provides a representative but significantly smaller subset of the original data and is designed so that the entire ML pipeline only takes a few minutes to perform its calculations.

This approach allows developers to use the development dataset for local development on their local systems. Additionally, the runner or the executing system can quickly provide feedback on whether a specific code change leads to a disruption or other problems in the ML pipeline.

The creation of this development dataset must be reproducible, version-safe and should therefore not be done manually. The ideal solution, as shown in Figure 4.1, is a standalone code module that draws samples from the original full datasets. This module should allow automatic updates of the partial datasets when the full dataset changes.

Assuming this code module offers the option to include specific samples or sample clusters in the development dataset, it could serve as a basis for regression tests. In these tests, samples that have caused problems in the ML pipeline in the past could be continuously monitored and examined from the beginning of model training to identify and fix potential error sources early. Additionally, implementing data quality checks at this stage ensures that the training and evaluation data meets predefined standards, reducing the risk of errors or biases. These checks, which, e.g., validate data types, detect missing values, and ensure proper data distribution, can catch anomalies early, preventing them from affecting the entire pipeline. Combined with data versioning, these checks offer a comprehensive view of data quality over time, enhancing the overall reliability of the ML pipeline.

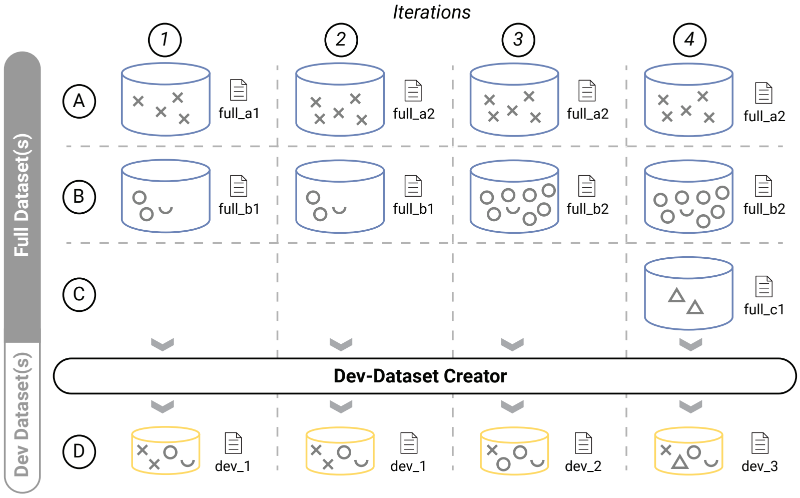


**Figure 4.1:** Development dataset concept. The dynamics of the complete datasets involved in the AL project are represented in separate iterations. The data within these iterations may change, or new data sources can be accessed. The Dev-Dataset-Creator module generates a small subset from this, which is provided as a developer dataset within the project.

Considering the dynamic nature of the data in data-centric AI projects, even more in AL projects, further requirements arise for this development dataset, shown in Figure 4.1. As seen from the change from the first to the second iteration, the Dev-Dataset-Creator module can mitigate the dynamics. Depending on the use case, the data basis is subject to permanent changes, e.g., due to stream data, which makes development with this data impractical, even if these were processed into batches. However, the development dataset should approximate the distribution of the entire datasets as accurately as possible, which occurs between the second and third iterations. If the distribution of the full dataset changes significantly, a new version of the development dataset is created automatically ($dev_1 \longrightarrow dev_2$).

Furthermore, outliers and distorted samples should be included from the beginning to ensure a realistic development environment. If new data sources are tapped during the project's duration, these must be considered in the development dataset, as seen in the fourth iteration.

Figure 4.1 illustrates a simplified outline. A project may require various development datasets that differ in size and content. It's important to understand that this dataset does not claim to serve for training a high-performance model, nor – contrary to common terminology – to test its performance during development time. As a primary target group of the development dataset, the team can use it during the development iteration to check the executability of the ML pipeline code on their local clients before their push and use it, e.g., for integration tests in a CI job, and therefore improve the quality of the committed code.

### 4.2.2 Code Principle

During initial phases, especially while prototyping, consolidating all elements of the ML pipeline into a single script or notebook is a widespread practice (Rule *et al.* 2018). Yet, such a method presents pitfalls. Not only are these artifacts typically constrained to a lone developer's workspace, but they complicate version-tracking. To foster effective collaboration within AI projects, it becomes imperative to disintegrate the ML pipeline's various steps into distinct modules, facilitating collaborative work amongst a development team (Amershi *et al.* 2019). This modularity enables team members to zero in on specific segments of the pipeline, optimizing work distribution.

The proposed project architecture, as depicted in Figure 4.2, underscores the alignment of individual code modules with the distinct phases of an ML pipeline. Crafting an ML pipeline mandates the discernment of requisite stages and the lucid articulation of inter-stage dependencies.

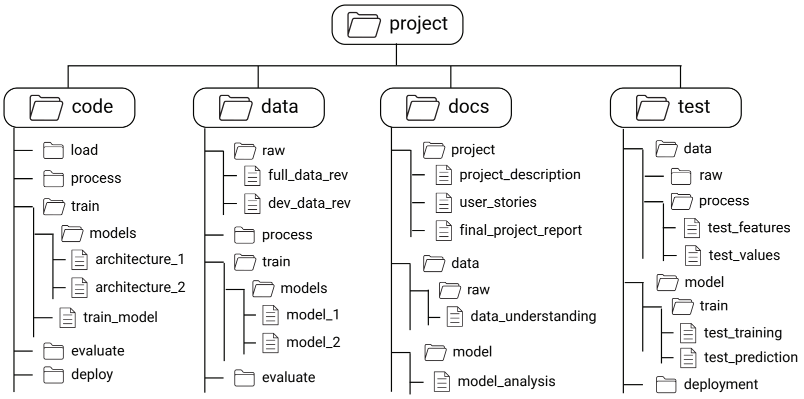


**Figure 4.2:** Proposed project structure. The elementary sections are structured into code, data, docs, and test.

As soon as an engineering team begins the implementation phase, it is essential to ensure that the data passes through the ML pipeline as early as possible, even if it has yet to produce meaningful results or artifacts such as trained models. The development dataset mentioned in Section 4.2.1 can be used to provide data to the code modules during the development phase. By ensuring this early data flow, developers can work simultaneously on different phases and manage their individual tasks.

Therefore, it is necessary to transform all steps of the ML pipeline into an automatically executable form. A sustainable code implementation, e.g., using parameters and configuration files, allows for different implementations and settings of each stage. By tracking these through a version control system like Git (Chacon *et al.* 2014) and using feature flags, a systematic comparison is possible even when multiple developers and data scientists work on the same ML pipeline stage.

This established software development concept allows specific features or changes in the code to be turned on or off during runtime without deploying new code. It enables different software engineers or data scientists to work on various features or settings in parallel without interfering with each other's work. In an AI project, this further facilitates controlled experimentation and rapid iteration, ensuring that new changes or optimizations can be evaluated and integrated seamlessly into the ML pipeline without disrupting the ongoing work of other contributors.

To meet further requirements regarding traceability, the „version number", e.g., the hash information in the tracked metadata file of the input dataset, must be treated as part of the configuration. Some data versioning tools, like DVC (Iterative 2020), offer this by storing hashes of the datasets as version numbers in the configuration files. These tools implement a Git-like architecture, where the concepts of Git version control are applied to data. Other approaches, like DeltaLake (Armbrust *et al.* 2020), utilize a multi-hop layer architecture, where data versions are created at each layer or „hop" in the data flow. In this approach, only the changes between versions are stored. Moreover, regardless of their architecture, data version control systems can often cache ML pipeline artifacts (e.g., preprocessed data, trained models, or evaluation reports). They can restore these artifacts if an execution with identical code and data occurred in the past.

This capability creates synergies in terms of the runtimes of the ML pipeline throughout the AI project, whether in the local development environment, experimenting with the entire dataset, or in the production environment (Steidl *et al.* 2023). For example, when new training data flow into the pipeline, the model can be retrained and, if necessary, optimized and evaluated. On the other hand, the computationally intensive preprocessing step can be repeated for only some of the datasets. By skipping this step, the new model can be quickly deployed.

### 4.2.3 Automation Principle

An important aspect in AI projects is the increased demand for infrastructure (Giray 2021; Ruf *et al.* 2021). This is reflected in the growing field of MLOps research, which encompasses managing different environments and effectively scaling hardware resources, along with the associated concepts and tools. Figure 4.3 outlines a simplified infrastructure setup agnostic to the underlying technologies and illustrates the necessary components for implementing a distributed system for AI projects.

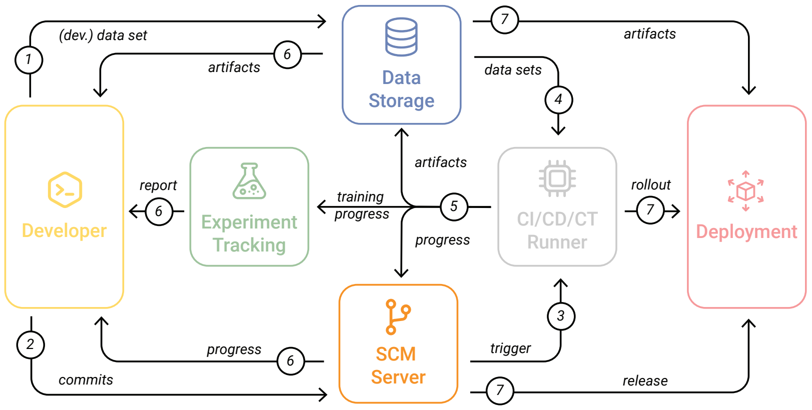


**Figure 4.3:** Concept of a minimal infrastructure setup. The different components and their relevance for realizing the illustrated workflow are shown.

In step ①, the raw data and the created development dataset must made available to the data storage. Typically, the upload is not done directly from the developer's client, but it is imported from the corresponding data source for larger datasets. The development dataset remains on the developer's client and should be small enough to enable fast iterations during ongoing work. Once the required datasets are available on the data storage, at step ②, developers commit and push their code to the Source Control Management (SCM) system. An SCM application is able to trigger a runner, the CI/CD/CT runner, at step ③. This runner should be hosted on a powerful machine (e.g., with GPUs) and takes into account the planning, management, and scaling of the computational resources needed for executing the ML pipeline. This becomes necessary when the resources exceed the capacities of the developer's computer, such as training the model on the entire dataset.

Each job is uniquely associated with a specific commit, facilitating easy decision-making on whether certain long-running jobs can be terminated. Existing tools and frameworks from traditional software projects can be reused to manage and monitor the servers where the runners are executed. In step ④, the runner checks out the version of the dataset specified in the configuration files and runs the ML pipeline.

In step ⑤, the runner reports the status to the SCM system and uploads artifacts, such as the trained model or preprocessed data, to the data. This can occur when the developer submits new code, conducts experiments on the entire dataset, or triggers an automated job, such as scheduled nightly retraining. The SCM application allows access to the runner's logs, providing the simplest solution for monitoring the progress of the ML pipeline. This can be further enhanced by using appropriate tools like MLflow (Zaharia *et al.* 2018), which enables the collaborating team of data scientists at step ⑥ to make structured comparisons of experiments or monitor the model in the production environment.

The presented infrastructure concept is minimalistic and can be expanded in different ways: Separate runners for CI, CD, and CT can be replaced by powerful clusters, e.g., the data storage can be enhanced by a feature store and/or a model registry. Additionally, the deployment implementation, as shown in step ⑦, can be further refined and is intended to indicate the interconnections of the individual components.

## 4.3 Branching Workflow

One of the core ideas of the proposed development approach for data-centric AI projects is to utilize the CI/CD/CT runners introduced in Section 4.2.3 for automation purposes using a branch-based workflow concept. This involves introducing different namespaces for branches, with each namespace focusing on either the code or data dimension. The runners behave differently based on the namespace of the branch in which they are triggered.

This organizational structure provides a structured environment for a team collectively working on an AI project repository. Furthermore, it serves as a catalyst for agile development by efficiently adapting the feedback cycles to the specific implementation types, whether application- or data-oriented tasks. To further investigate the branching workflow, we will illustrate the concept through an example project in the subsequent Section.

### 4.3.1 Main Branch

After the scoping phase is completed and the setup described in Section 4.2 is in place, the code for the initial ML pipeline is first transferred to the repository's main branch in the SCM application. Similar to traditional software projects, the main branch, contains a complete version of the code. Because the main branch is characterized as long-term, it is advisable to follow the proposed code structure from the beginning, even if, in the initial steps, the focus should still be on the exploratory phase of the data. At the beginning of an AI project, the rest of the code may consist solely of stubs and interfaces for each downstream phase of the ML pipeline. Subsequently, the primary requirement for the main branch is to provide a clean, executable, and stable version of the ML pipeline.

In this regard, the focus of the CI/CD/CT runners in its branch namespace lies on code quality: they execute the ML pipeline using the development dataset as a form of integration testing. Given that the configuration files contain information about the allowable values of all parameters (cf. 4.2.2), they can verify the code across the permissible combinations, which, even with the development dataset, can prove to be time-consuming.

Furthermore, traditional unit tests and other code analysis steps should be performed to maintain the desired high quality in the main branch. Similar to traditional software development practices, static analysis tools can be employed to uncover potential issues in the code, such as poor programming practices or unused variables.(Lenarduzzi *et al.* 2021)

These additional steps contribute to the early detection of potential errors and ensure a robust codebase for the ML pipeline. By incorporating quality assurance aspects, all team members are empowered to address potential issues and provide smooth executability of the ML pipeline at all times.

### 4.3.2 Feature Branches

As soon as the development team envisages a new feature[3] or the need arising from altered requirements, a feature branch is created, adhering to the established concept of traditional software projects (Chacon *et al.* 2014). Usually, this need is documented in an issue within the SCM application, and the feature branch, exemplified in Figure 4.4 by commit A1, is generated. The branch where the implementation takes place is at the level of code-focused branches. In this process, the developer generates a new feature flag along with the necessary parameters and implements the new feature, utilizing the development dataset.

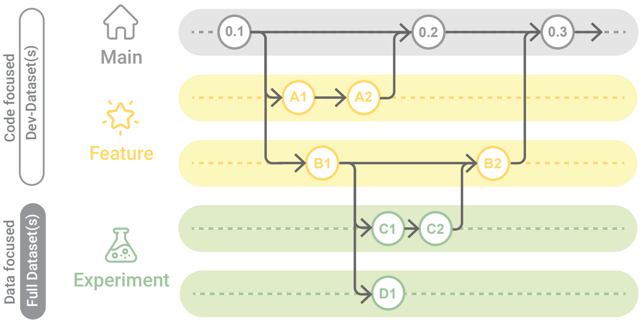


**Figure 4.4:** Git workflow with Experiment-Branches.

After every push commit, the runner verifies the ML pipeline's correct executability, checking out the development dataset for the code execution. Analogous to a CI pipeline in classic software development, static code checks and unit tests are conducted here as well. The rapid executability using the development dataset provides the author with swift feedback about the success of the CI jobs. However, the training results of the ML pipeline are irrelevant. Hence all services used in the project to track experiments remain deactivated.

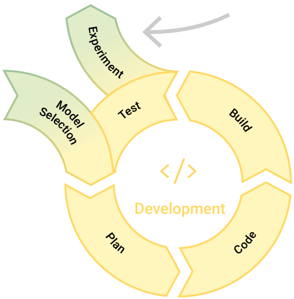
Once the developer is convinced that the problem described in the issue has been solved, they initiate a merge request. Such requests enable other team members to provide feedback and review the code before it flows into the main branch and morphs from commit A2 into Version 0.2. While the focus of the feature branch remains code-centric, data-related considerations – such as improvements in model performance or data quality tests – are handled separately in a dedicated branch namespace, which will be discussed in the following section, ensuring that these aspects are addressed before the final merge.

As illustrated by the example of the first feature branch, it can sometimes suffice to merely implement and test the function, enabling other team members to build upon the latest features as quickly as possible. In certain cases, combining various features might be necessary to genuinely enhance the ML model's performance. As the project follows the code structure outlined in Section 4.2.2, it is entirely conceivable that several developers are simultaneously implementing different functions in various branches.

Sometimes, the implementation of certain functions in an AI project necessitates their execution on the entire dataset, and this even before the implementing team member initiate a merge request. This case is exemplified in Figure 4.4 within the second feature branch B1. In this

[3] „Feature“ in this context follows the SE terminology of Lenarduzzi *et al.* 2021: „A distinguishing characteristic of a software item“

scenario, the developer decides to initiate an experiment and branches out with the corresponding code version into one or more experiment branches.



Related phases in the AL development life cycle: The experiment stage links the development iteration to the ML iteration.

### 4.3.3 Experiment Branches

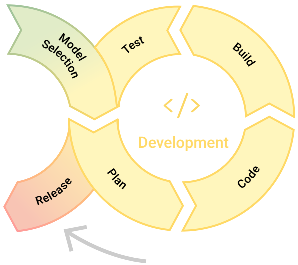
Certain tasks in an AI project are experimental in nature (cf. 3.3.2). While the widespread way of implementation for data science related jobs takes place in notebooks (Rule *et al.* 2018), in the proposed concept a custom branch namespace will be introduced. This special category of branches, called „Experiment Branches", runners execute the ML pipeline on full datasets and not only on development datasets as is traditionally the case. This expands the testing space and potentially provides more representative results.

An exemplary situation is the implementation of a new model architecture by a data scientist. The executable code is implemented in the feature branch (B1). The next step would be to train and evaluate the new model's performance. This is where the experimental branch becomes relevant.

The data scientist creates one or more branches in the experiment namespace and configures the ML pipeline according to the specific needs of the test. This is done by enabling the feature flag and setting appropriate parameters in the configuration files. Each of these experiment branches can focus on specific aspects of the ML pipeline. For example, one experiment might be specifically reserved for hyperparameter optimization, while another might enable additional data augmentation.

Once a configuration is set, for example, as illustrated in the commit (C1), this triggers the runner to access the full datasets. If the ML pipeline execution is complete, the runner stores the artifacts on the data using the data versioning system. Consequently, it is ensured that all boundary conditions – from the data version to the code version to the execution environment – are documented for each artifact.

This concept can additionally be used for feasibility experiments. These provide the developers with preliminary evaluative feedback regarding the feasibility and potential efficacy of the current function under consideration. In such cases, it might be helpful to trigger proof-of-concept experiments. In doing so, the focus shifts from fine-tuning the model to roughly evaluating ML performance. If an experiment fails, as visible in the example of commit (D1), the overhead of a merge request and code reviews is avoided. This saves downstream resources and allows for more efficient use of development time.



Releasing a version exits the development iteration.

### 4.3.4 Release Branch

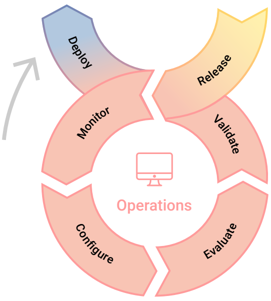
Suppose the results of an experiment outperform the current leading model or a completed feature is to be released. In that case, merging from the main branch to the version branch is initiated. Throughout this phase of the model life cycle, the implemented code as well as the produced artifacts, are subject to thorough inspection and assessment by additional team members.

In the scenario, illustrated in Figure 4.5, the acceptance of the merge request of version 1.0 triggers the activation of a new runner on the release branch. Due to the caching feature of a data versioning tool, this runner can restore and reactivate the artifacts from an experimental run on the data storage. Should errors or undesired states occur during or after the deployment, proven, corrective measures from traditional software development can be applied, such as reverting the release branch to an earlier commit. The same applies to model versions in case of a later concept deviation. The release branch provides traceability for the entire development team, allowing recourse to already released versions of a model registry.

We establish an additional specialized branch type to accommodate the continuous dynamics of data and code within a data-centric AI project. This new branch exists on the data-focused layer of the project and serves as a code-based version twin of the release branch. This is of particular importance as it allows maintaining synchronized versioning of code and data, which is essential for the traceability and repeatability of experiments. To make the concept more concrete, we continue the example, where the development team decides to go live with version 1.0.

### 4.3.5 Production Environment

The branch referred to as production branch is defined as the mirrored, deployed code version of a branch that references a possibly newer version of the corresponding data artifacts. This state is achieved by the consistent use of runners that, in the context of continuous training, are used to transfer the data artifacts to the data storage after execution of the ML pipeline and to transfer the version reference of the hashes back to their production branch. Depending on the purpose of the application, this can result in this branch potentially running ahead of all branches under development.



Upon the team's consensus to transition a version to a production environment, this action aligns with the deployment stage of the AL development life cycle.

Build upon the example of experiments presented in Section 4.3.3, focusing on the colored blue branch and demonstrating that the experimental branches can be used to reconfigure the ML pipeline. This approach is evident in Figure 4.5, where the development team directly branches off a new branch E1 from the main branch into the experiment namespace to modify the configuration file. Now, we utilize the previously mentioned synergy between the caching function of the data versioning tool and the reuse of artifacts in the other direction. When a runner is triggered in an experiment branch, it can fetch the current version of the data reference file from the production branch and download all existing data artifacts from the data storage.

The code of the experiments will be executed with the actual data, providing the development team the opportunity to make decisions based on consistently traceable and reproducible results. As mentioned, runners triggered by the transmission of an experiment branch should, by default, activate an experiment tracking service. This workflow makes the development team's decision to reconfigure transparent and reproducible at any time.

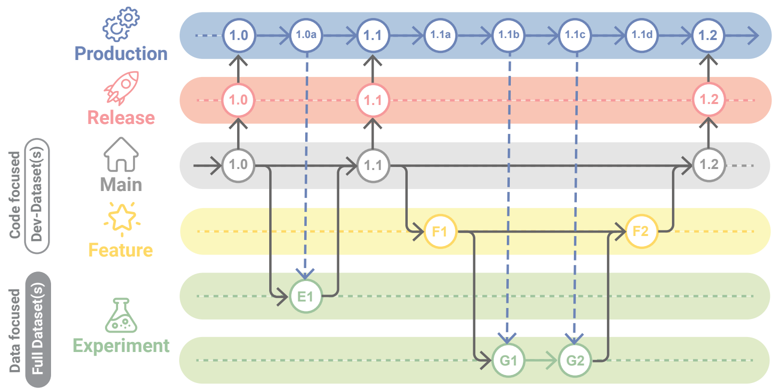


**Figure 4.5:** Git workflow with production environment

Implementing simulations in the production environment takes it further, especially in applications with AL or online ML. Here, it is not enough to switch from development datasets to complete datasets. Rather, the runner of the experiment branches must be able to verify the updated, published data version. This process is depicted from commit F1, where the development team, for instance, implements a new model architecture.

The data scientist initiates a branch at G1 to evaluate performance under production conditions into the experiment namespace. This time, an automation is activated, ensuring that the runner always uses the latest version from the production environment. This mechanism allows for realistic tests with current data, thus providing immediate insight into the performance of the current system version under real conditions. The experiment branch can coexist parallel to ongoing development. Suppose the team decides to make further changes to the code. In that case, the branching workflow remains the same: a merge request is derived from F2, and the newer version of the ML configuration is propagated through the merge in the main branch as version 1.2, which is provided via the release branch in the production branch.

## 4.4 Git Workflow for Active Learning

Implementing an AL project underscores the dynamic nature of code, data, and model since the continuous iteration of retraining and data annotation results in a fast-paced character of the resulting artifacts. The concepts presented in Section 4.2 can be applied to AL projects, and the branching workflow shown in Section 4.3 can be adapted.

To illustrate this, Figure 4.6 depicts the workflow for an AL project in which the production branch has been renamed to the „AL Branch“. It represents the version twin of the currently deployed model. This AL Branch is a direct reflection of the deployed branch but with a critical difference: it integrates the latest data version containing new annotations.

To achieve this, we deploy runners in our CI pipeline. Yet, these runners manage two essential functions:

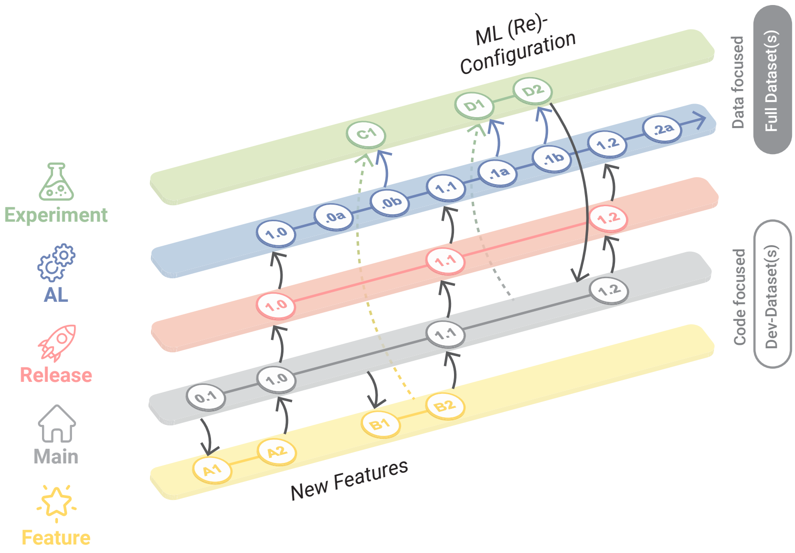


**Figure 4.6:** Git workflow for an active learning project. The AL Branch represents the production environment. While the runners in the code-focused branches check out the dev dataset, the execution of the ML pipeline in the data-focused branches will be performed on the full datasets.

- They transfer updated data artifacts — those enriched with the latest annotations — into data storage upon completion of the ML pipeline.
- They update the version reference of the data artifacts, represented as hashes, to reflect these changes in the production branch.

Again, this approach ensures that the production branch has a lead in data versions over all branches that are still in development, according to the application's requirements. Moreover, this separation between the AL Branch and the release branch not only allows a structured process for integrating new software features but enables the development team to roll back to a previous model version if there is a concept drift or in the event of a deterioration in performance.

Furthermore, the workflow depicted in Figure 4.6 clarifies the concept of code- and data-focused branches. While the workflow for feature branches Ⓐ remains as described in 4.3.2, the example of feature branch Ⓑ could now correspond to implementing a new QS. The developer branches from feature branch Ⓑ to an experiment branch Ⓒ, where the runner checks out the current version of the annotated dataset to evaluate the QS in a simulation.

Suppose the development team wants to test a QS or the performance of automated data labeling in parallel with regular production operations. In that case, this is possible using the live experiments described in 4.3.5. For this purpose, a data scientist can branch off an experiment branch Ⓓ and make the respective configurations for the selection strategy or semi-supervised learning. In the example from Figure 4.6, the data scientist decides again to incorporate the current data from the production environment into his experiment. This allows for realistic testing with existing data and provides immediate insights into how the model would behave under live conditions. Again, the example shows how the potential merge into the main branch manifests if the configuration made in the experiment leads to the desired behavior of the model.

This procedure fulfills two essential properties concerning transparency: first, the experiments conducted are comprehensible for the entire development team. Second, the reproduction of results is possible at later points in time, even if the data situation has changed again due to the advancing project process. Regarding the working method, the proposed workflow additionally offers the advantage that the experimental tasks of the development teams are structured, which is particularly relevant for more extensive projects since the results of experiments already carried out may be visible and reusable for other team members.

## 4.5 Evaluation

The proposed process methodology is evaluated according to the three-phase concept outlined in Figure 4.7. Both the theoretical orientation and the practical suitability are considered.

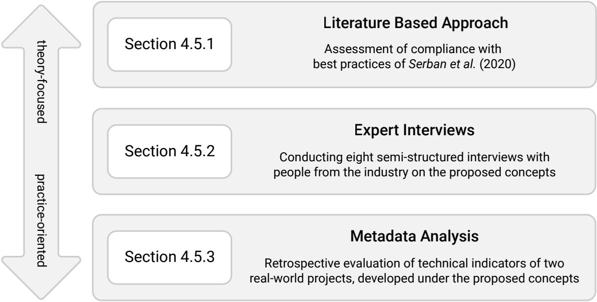

theory-focused
practice-oriented
Section 4.5.1
Literature Based Approach
Assessment of compliance with best practices of *Serban et al.* (2020)
Section 4.5.2
Expert Interviews
Conducting eight semi-structured interviews with people from the industry on the proposed concepts
Section 4.5.3
Metadata Analysis
Retrospective evaluation of technical indicators of two real-world projects, developed under the proposed concepts

**Figure 4.7:** Evaluation concept for the engineering methodology proposal

To assess the practicality of the introduced basic concepts and workflow, the evaluation of the proposed methodology occurs in three phases. In the initial phase (4.5.1), a comparison with existing literature is conducted, providing a benchmark against best practices. For the secondary assessment (4.5.2), the results of interviews are utilized to bring in a practice-oriented evaluation from industry experts. The tertiary step (4.5.3) involves the metadata analysis of two projects in which the methodology was applied.

### 4.5.1 Literature Based Approach

To evaluate the proposed principles and the branching workflow, we rely on the best practices collected from literature by Serban *et al.* 2020. The authors focus on peer-reviewed publications that suggest, compile, or validate proven SE practices for ML. Their method resulted in 29 best practices that are categorized into data, training, coding, deployment, team, and governance.

We will briefly discuss the catalogue of practices compiled by Serban *et al.* 2020, categorizing the extent to which the proposed engineering process methodology is capable of integrating the identified best practices and the degree they are inherently supported (through categorisation: *fulfilled*, *achievement supported*, and *not prevented*). For each best practice identified, we will first name and outline the practice itself, followed by an discussion evaluating how well the methodology addresses it. Suppose a best practice is not yet fulfilled per se. In that case, we will explore the necessary prerequisites that teams need to meet to effectively apply the principles and procedures within the concept, identifying areas for potential improvements in the process.

**Data.1**

Achievement supported by 4.2.1

**Use sanity checks for all external data sources.** All data sources incorporated into the AI project should be systematically checked for plausibility and correctness. These checks aim to identify and rectify errors, inconsistencies, or quality issues in the data at an early stage before they are used for model training or inference. This ensures that the ML model is built on reliable and valid data, which is a pivotal prerequisite for the performance and reliability of the resulting models.(Polyzotis *et al.* 2018; Breck *et al.* 2019; Baylor *et al.* 2017; Google 2023b)

The implementation in practice involves various steps. Live logging and monitoring can set up automated testing routines and define data validation rules. Referring to the stated requirements for the development dataset (cf. 4.2.1), the described module of an Dev-Dataset Creator could encompass this logic and cover checks e.g., for missing values, unexpected formats, value ranges, or statistical anomalies, for example. As described, this process is ideally automated and regularly imports all data sources. Assert statements that check for thresholds or permissible formats can be integrated into this module. The concept described can therefore have a positive effect on the fulfillment of this best practice during the development phase. However, further steps are necessary for full implementation, particularly in the application to be implemented.

**Data.2**

Achievement supported by 4.2.1

**Check that input data is complete, balanced and well distributed.** This practice aims to ensure that the data used to train the model are complete and represent a representative sample. Therefore, the data should not have biases, such as by having an even distribution of classes in classification tasks, and should adequately reflect the variability of the real-world context for which the model is being developed.(Baylor *et al.* 2017; Breck *et al.* 2017b; Breck *et al.* 2017a; Polyzotis *et al.* 2018; Sculley *et al.* 2015)

In gerneral, different measures are necessary to implement this practice. Through data exploration and analysis, the distribution of data is reviewed (cf. 3.3.2). Furthermore, data preprocessing related tasks contribute to this, for example, fixing potential imbalances in the data (cf. 3.3.2). Again, the fulfillment of this best practice is not inherently ensured by the application of the presented methodology. Nevertheless, the requirements for the development dataset mentioned in Section 4.2.1 contribute to the engineering team's continuous engagement with data completeness and distribution since the development dataset provided for implementation and in the code-focused branches mirrors the distribution of the full datasets, including potential faulty data. This approach encourages the team to develop an awareness of dealing with the given data quality during the implementation phase.

**Data.3**

Fulfilled by 4.2.1, 4.2.2

**Write reusable scripts for data cleaning and merging.** Standardizing data cleansing and merging processes is recommended to ensure these tasks are performed efficiently, consistently, and transparently. This is particularly important when scaling projects involving multiple developers and when data processing steps occur at various points within the project (Polyzotis *et al.* 2018). We consider this best practice to be fully met by adhering to the presented data- (4.2.1) and code principles (4.2.2).

**Data.4**

Achievement supported by 4.2.1, 4.6

**Ensure data labeling is performed in a strictly controlled process.** Requirements are typically set for the data annotation process, such as implementing quality controls (Roh *et al.* 2021). This entails procedures and guidelines to ensure the accuracy and precision of the annotations that are used for training supervised ML models. Controlled processes may include, for example, regular checks and multiple annotations to ensure the reliability of data labels (Alonso 2015; Bernhardt *et al.* 2022).

Although many of these aspects are examined in greater depth from the system perspective in Chapter 5, the proposed SE process is an enabler to fulfill this established best practice. Following the data principles (4.2.1), the consistent versioning of all artifacts positions the development team to ensure the necessary consistency, such as implementing the process of multiple annotations. Through the introduction of an AL loop and implementation using the AL branch, this development process can be traced throughout an AL project (4.4).

**Data.5**

Achievement supported by 4.2.1, 4.2.3

**Make datasets available on shared infrastructure (private or public).** This pertains to data storage on a shared infrastructure, which enables efficient resource use, enhanced collaboration, and more accessible data exchange. Data should be stored on platforms available to all involved developers, adhering to security and privacy regulations. This approach facilitates data management and supports the reproducibility of artifacts by allowing project team members to access existing datasets or previously created artifacts such as extracted features or trained models (Khomh *et al.* 2018; Zha *et al.* 2023). By adhering to the proposed automation principles (4.2.3) and the infrastructure outlined in Figure 4.3, this best practice is fulfilled.

**Training.6**

Achievement supported by 4.3.3

**Share a clearly defined training objective within the team.** Emphasizing the significance of clearly defined and communicated training objectives for the ML model within the team stands at the core of this best practice. All participants, including those in non-technical roles, should have a common understanding of the specific outcomes and performance indicators to be achieved by the ML model. This clarity enhances the alignment of activities and contributes to decision-making, which ideally is supported by the entire team, thereby improving collaboration and ensuring that the results achieved are in line with the goals defined for the project.(Lazzeri 2019; Zinkevich 2019)

We do not see this best practice as inherently fulfilled by following the introduced engineering process; however, the concepts presented contribute positively to enhancing communication within the development team and providing the necessary traceability to document training progress. This is particularly evident concerning the experiment branches (4.3.3) and the recommendation to use an experiment tracking service in the data-focused branches, enabling continuous review of the training objectives.

**Training.7**

Achievement supported by 4.2.2, 4.3

**Capture the training objective in a metric that is easy to measure and understand.** To objectively assess the progress of the model's learning process, it should align with the practice of clearly defined training objectives, which are concretized in a performance metric for evaluating the ML model. This metric should be both easy to measure and straightforward to understand. The choice of metric significantly depends on various factors, but primarily on the type of problem, the algorithm aims to solve. To regard this best practice as fulfilled, additional efforts should be made to make the results, for instance, in dashboards and reports, easily interpretable.(Lazzeri 2019; Microsoft 2020; Zinkevich 2019; Google 2023b)

While the concepts presented have no direct influence on this, we argue that the project structure introduced in Section 4.2.2 ensures the implementation of a uniform evaluation process over the entire project. This ensures measuring the model's performance is standardized and transparent for involved team members when using the proposed branching workflow (4.3) and experiment tracking services.

**Training.8**

Achievement supported by 4.2.1, 4.2.3
Enhance 4.3.1, 4.3.2

**Test all feature extraction code.** This best practice underscores the necessity of thoroughly testing the code for extracting features[4] from raw data. Since the quality and reliability of feature extraction directly affect the performance of the ML pipeline, this code must function correctly. Tests should ensure that the corresponding functions and modules lead to consistent and accurate results under various conditions and are robust against input errors and data anomalies.(Breck *et al.* 2017b)

As described in Section 4.2.1, the proposed development dataset can serve as a basis for regression testing, which the development team can expand to outline the feature extraction code. Furthermore, suppose unit tests are implemented for this code. In that case, the proposed concept provides for the CI runners to automatically execute these routines on feature- and main branches upon every commit (cf. 4.3.1, 4.3.2), ensuring the correct and robust operation is verified even on edge-case samples in the development dataset. Additionally, having the development dataset available in the local development environment encourages the team to implement the necessary test routines, as they are urged to let the data flow through all steps of the ML pipeline throughout the entire development life cycle.

**Training.9**

Fulfilled by 4.2.1, 4.2.2

**Assign an owner to each feature and document its rationale.** Every feature from the dataset should have an assigned owner responsible for its development. The logic, including the rationale for selecting a feature and the method of its calculation, should be meticulously documented. Adhering to this best practice enhances traceability, accountability as well as transparency and simplifies the management of features throughout the life cycle of the ML model, as each feature has clearly defined points of contact and documentation. The consistent use of code- and data versioning, along with implementation in feature- and experiment branches, ensures the fulfillment of this best practice.(Zinkevich 2019)

**Training.10**

Achievement supported by 4.2.2, 4.2.3

**Actively remove or archive features that are not used.** In AI projects, complexity reduction of the model and more efficient resource use can be achieved by cleaning and maintaining feature sets, which includes the removal or archiving of features that have become unused or irrelevant. Furthermore, a tidy feature space can improve the interpretability of the model.(Sculley *et al.* 2015; Zinkevich 2019)

To achieve this best practice, further techniques are usually necessary, which can be integrated into the presented concepts. For instance, if feature importance techniques are implemented in the automated pipeline,

[4] „Feature" in this context follows the ML terminology of Lenarduzzi *et al.* 2021: „A characterizing variable found in the input data to a ML model. Predictions about the data can be made after gaining insight from these features (training)."

these routines will be executed in the CI runners, facilitating regular reviews by the development team. The modularization of the ML pipeline, as well as the versioning of artifacts, contribute to making an informed and traceable decision for the team regarding the features used and documenting this process.

**Training.11**

Fulfilled by 4.2.2, 4.3

**Peer review training scripts.** Within the process of AI projects, training scripts that are used for the development of ML models should be reviewed by one or more additional team members before being deployed to production (Breck *et al.* 2017a). This practice aims to detect errors early, enhance the quality of the code, and ensure that the methodology and logic behind the training process are correctly implemented. Moreover, a peer review process encourages knowledge sharing and collaboration within the developer team. We consider this best practice inherently fulfilled in reference to Sections 4.2.2 and 4.3 of the proposed engineering method.

**Training.12**

Fulfilled by 4.2.3, 4.3.3

**Enable parallel training experiments.** The provided platforms should enable the team to simultaneously run multiple training experiments for models. This allows for the testing and comparison of various hypotheses and configurations at the same time and enables the parallel implementation of the project and the modules of the ML pipeline by several team members. Both aspects contribute to enhancing the efficiency of the development process.

We consider this best practice fulfilled as described in Section 4.2.3 and by managing experiments in separate branching namespaces (4.3.3), assuming that the provided infrastructure resources have the necessary capacities.

**Training.13**

Achievement supported by 4.2.3
Enhance 4.3.3, 4.3.4

**Automate hyper-parameter optimisation and model selection.** This best practice focuses on automating processes related to hyperparameter tuning and thereby selecting the best model. Typically, techniques are used that efficiently determine the best combinations of hyperparameters to improve the model's predictive performance.(Hutter *et al.* 2019)

It is not inherently fulfilled, but the proposed workflows can support its implementation. On the one hand, using different CI/CD/CT runners ensures that optimization routines are executed only when, for example, an experiment is launched, or the development team decides to release a new model version. Another aspect is that redundant hyperparameter optimizations can be skipped, which, considering the typically computationally intensive nature of these processes, can significantly impact the project's productivity. Using a SCM application in combination with a data versioning tool, the shared data storage, as well as different runners for CI, CD, and CT, ensures that already computed artifacts are not recreated and are loaded upon re-execution in the ML pipeline (cf. 4.2.2). This approach ensures that existing infrastructure resources are used efficiently.

**Training.14**

Fulfilled by 4.3.5

**Continuously measure model quality and performance.** Trained models should be regularly analyzed, and performance metrics measured. This continuous evaluation of quality and performance contributes to identifying potential issues and enables the engineering team to make adjustments to ensure the desired functionality during development and after deployment.(Zinkevich 2019; Google 2023b)

The concepts described a necessary minimum to fulfill this best practice. As mentioned in 4.3.5 the experiment tracking service is enabled by default in the CI/CD/CT runner used for the production branch. This ensures that each execution of the ML pipeline in the production environment provides the development team with newly calculated performance metrics. However, this can be supplemented with advanced monitoring tools for even more enhanced performance tracking.

**Training.15**

Fulfilled by 4.3.3

**Share status and outcomes of experiments within the team.** In large AI projects with multiple team members, as in traditional software projects, the formation of knowledge silos should be avoided. This is particularly true for the experiments conducted. Transparent communication about the status and outcomes of experiments promotes collaboration. It allows team members to learn from each other and helps ensure that decisions are made based on complete and up-to-date information.(Lazzeri 2019)

With the proposed experiment branches and the use of runners that transmit results to the experiment tracking service, we consider this best practice to be fully met. Even though this best practice is technologically fulfilled through the collaborative development process, it should encompass the necessary communication culture regarding experiments, which could be supplemented by additional appropriate measures such as regular meetings.

**Training.16**

Fulfilled by 4.2.1, 4.2.2

**Use versioning for data, model, configurations and training scripts.** A crucial aspect of the reproducibility of results and tracking all artifacts created during an AI project is managing relevant code components, including datasets, models, and configuration files, using version control systems (Hummer *et al.* 2019; Van Der Weide *et al.* 2017; Washizaki *et al.* 2019). Furthermore, this approach contributes to creating a transparent working method and provides an organizational basis for meeting goals related to accountability by documenting who made which changes and when. This best practice is met by the concepts presented, particularly in Section 4.2.1 and 4.2.2.

**Coding.17**

Achievement supported by 4.2.1, 4.2.3

**Run automated regression tests.** The automated execution of regression tests is designed to ensure that new changes in the code, data, or model do not negatively affect the existing functionality of the AI system. This procedure integrates continuous checks to see if updates or improvements cause unexpected errors or performance declines. As with traditional software projects, this consistent quality assurance contributes to maintaining the system's stability over time.(Breck *et al.* 2017b; Google 2023b)

Implementing a comprehensive testing concept for AI projects involves a multitude of steps, such as integrating a test plan and creating test cases for the ML pipeline, which concern data, code and the model itself. Therefore, merely establishing the proposed development process does not imply fulfilling this best practice. However, the concepts again contribute to its implementation. As described in Section 4.2.1., the automatically created development dataset can be used in runners to perform regression tests. This strategy promotes the implementation of test cases with the data flow through the ML pipeline, as the potentially high number of possible permutations of ML pipeline executions can be realized in the regression tests using the development dataset. Additionally, the synergies of combining SCM and data versioning tools regarding their artifact caching can be utilized for these tests, where previously tested combinations of configurations and data and code versions can accelerate the involved routines.

**Coding.18**

Fulfilled by 4.2.3

**Use continuous integration.** The process of continuously merging code changes into a shared repository, followed by automatic builds and tests, is a practice from the DevOps concept. In AI projects, continuous integration includes source code integration and data and model changes (Breck *et al.* 2017b). This process promotes collaboration within the team and thus helps maintain high software quality. This best practice is fully met by using CI runners as an integral part of the automation principle (cf. 4.2.3).

**Coding.19**

Achievement supported by 4.2.2

**Use static analysis to check code quality.** Static code analysis tools systematically check the quality of the source code without executing it as a program. This method makes it possible to identify issues such as syntax errors, stylistic inconsistencies, potential anomalies, or even security vulnerabilities. In large software projects, static code analyses help improve the code's clarity, maintainability, and reliability, which is particularly important for the code of the ML pipeline given the often complex data processing and modeling tasks.(Gomes *et al.* 2009)

We align with the argumentation of the best practice Coding.17, which states that while the introduced concepts may not cover the tasks of static code analysis, they do support its implementation in the project. Particularly, the code structure proposed in Section 4.2.2 highlights the necessity of using such tools, for instance, to ensure a consistent coding style across the various components of the ML pipeline. As described in Section 4.3.2, static code analyses can be executed at every commit triggered in a CI runner, providing developers with feedback on the status of their code changes. Depending on the convention, successfully passing this code analysis can be a mandatory condition for a merge request into the main branch, ensuring the desired high quality remains in the main branch.

**Deployment.20**

Fulfilled by 4.2.3, 4.3

**Automate model deployment.** Deploying the trained ML model into the production environment should be automated. This includes using tools and techniques that allow for the development team's rapid, efficient, and minimally manual transfer of models into operational use. From a process perspective, such automation avoids delays, contributing to the high readiness and scalability of the AI system (Arnold *et al.* 2020). Additionally, this approach moves towards end-to-end reproducibility (Van Der Weide *et al.* 2017). This best practice is fulfilled by following the described principles in Section 4.2.3 and the proposed branching workflow (cf. 4.3).

| **Deployment.21** |
|---|
| Achievement supported by 4.3.5 |

**Continuously monitor the behavior of deployed models.** Once trained ML models are deployed into the production environment, it is a best practice to monitor them to assess their behavior and performance continuously. This monitoring typically includes capturing metrics related to model performance, such as the error rate on data in the real-world environment (Baylor *et al.* 2017). As described in Section 3.3.4, continuous monitoring is crucial to promptly respond to issues like data drift, model obsolescence, or changing requirements.

This best practice significantly overlaps with the aspects described in Training.14. While continuous measurement of model quality and performance primarily focuses on the training process of ML models, the continuous monitoring of the deployed model's behavior shifts, for example, to emerging concept drifts. In practice, it is advisable to use comprehensive methods and implement them using powerful monitoring tools, where the concept of a separate production branch (cf. 4.3.5) can serve as an enabler.

| **Deployment.22** |
|---|
| Achievement supported by 4.3<br>Enhance 4.3.5 |

**Enable shadow deployment.** This best practice refers to a technique where a specific version of the software and/or model is operated in parallel to an existing version in production, without actually intervening in the deployment process. In AI projects, for instance, the new model version receives the same input data as the currently deployed model. However, the predictions and performance are only recorded and analyzed, not used for the actual action of the AI system. This method allows for testing and evaluating the performance and stability of a new release candidate under real conditions without affecting the running system or user experience.(Olston *et al.* 2017; Baylor *et al.* 2017; Van Der Weide *et al.* 2017; Washizaki *et al.* 2019)

Implementing this best practice requires further steps at the system level, such as separate logging and the integration and evaluation of necessary feedback loops. However, from a process perspective, this can be achieved with an additional, separate production branch. This branch adheres to the concept of shadow deployment and does not impact the running system but rather represents an additional mirrored twin of the release branch. Unlike the production branch or a long-lived experiment branch, e.g., used for live experiments, the data streams in this new production branch are used solely for observation and analysis.

Existing logic and tools for performance measurement, such as the experiment tracking service, provide the opportunity to compare the performance of models in different branches. If the results are satisfactory, a gradual transition can be implemented by merging the desired (model-) version in the described workflow from Section 4.3.4 from the release- to the production branch. This approach allows for careful monitoring and evaluation of the new version in a real-world environment without disrupting ongoing operations, ensuring a risk-aware transition.

**Deployment.23**

Achievement supported by 4.3.3
Enhance 4.3.3

**Perform checks to detect skews between models.** To identify discrepancies or shifts (known as „skews“) between training and production models, it is best practice in AI projects to regularly review any differences between the training data and real-world data from the production environment. This approach enables early detection of data shifts, overfitting, or model obsolescence, thereby positioning the team to take appropriate countermeasures, such as retraining the model.(Baylor *et al.* 2017; Zinkevich 2019; Google 2023b)

Again, the close connection with the best practices Training.14 and Deployment.21 is evident. Unlike best practice Deployment.21, which aims to assess whether models already released in the production environment are functioning as expected and thus considers the occurrence of anomalies and errors, the review of skews between models explicitly attempts to identify potential inconsistencies between training models and the production model. To fulfill this best practice, the described concepts of experiment branches, described in Section 4.3.3 can be used. They allow, using the experiment tracking service and the calculated performance metrics of the model version from the release branch, to compare the model performance of a newly trained model with the currently deployed model in the production environment in a traceable way.

**Deployment.24**

Fulfilled by 4.3.4

**Enable automatic roll backs for production models.** Implementing mechanisms that enable automatic reversion to a previous version of an ML model empowers the development team to respond swiftly, for instance, if the currently live model exhibits unexpected issues or performance drops. This best practice mainly contributes to the stability and maintenance of the reliability of AI systems (Zaharia *et al.* 2018). Automatic rollbacks ensure that in the event of emerging errors or problems, a switch can be promptly made to a proven, stable model version, thereby minimizing potential negative impacts. This best practice is fulfilled by the proposed branching workflow and the characteristics of the release branch, which allow for seamless transitions back to earlier, stable versions of the model when necessary.

**Deployment.25**

Achievement supported by 4.2.1, 4.2.2, 4.2.3, 4.3.5

**Log production predictions with the model's version and input data.** Detailed logging of the predictions made by an ML model in the production environment, along with information about the model version and the input data, enables comprehensive traceability of model decisions. This approach is fundamental for analyzing model performance and potential troubleshooting, as well as for complying with regulatory requirements, and it forms the basis for the team to understand the causes behind specific predictions made by the model in the production environment.(Hummer *et al.* 2019; Sridhar *et al.* 2018)

Typically, explicit steps are necessary on the system side to fulfill this best practice. In addition to implementing a comprehensive logging system, integrating the required logic into the ML pipeline and the structured metadata storage are prerequisites. Nevertheless, the concepts described from a process perspective contribute in different ways to fulfilling this best practice. First, the consistent versioning of data and code versions is an integral part of the methodology. Secondly, the described infrastructure provides the basis for storing the necessary prediction logs (cf. Section 4.2.3). The dedicated production branch, as described in Section 4.3.5, can serve as the starting point for logging the version of the model and input data.

**Team.26**
Achievement supported by 4.3

**Use a collaborative development platform.** This best practice focuses on using platforms and tools that support and enhance collaboration within the development team. Such platforms, often provided by cloud service providers, offer the necessary toolchain comprising version control, collaborative code editing, issue tracking, CI/CD processes, and experiment tracking services. This approach targets coordinated and collaborative work on a project, as well as communication and technical knowledge exchange (Booch *et al.* 2003; Storey *et al.* 2017). As mentioned in Section 4.2, the described concepts are agnostic regarding the choice of tools. Using the defined concepts and workflows takes account of collaborative cooperation as the core idea of this best practice.

**Team.27**
Achievement supported by 4.3

**Work against a shared backlog.** It has proven effective in software projects for developers to access and work from a shared list of tasks, known as a backlog. This backlog contains all planned features, improvements, bug fixes, and other tasks relevant to the project. This approach aims to ensure that all team members work towards the same objectives, with clearly defined priorities and a transparent and efficient workflow.(Sedano *et al.* 2019; Schwaber *et al.* 2020)

Although the described development methodology lays the foundation for working with a shared backlog, the team culture contributes a significant part to fulfilling this best practice. The defined branching workflow can promote fulfilling this best practice, especially when branches are linked to an issue, as recommended, and integrated into the backlog to provide a clear overview of pending tasks and their status.

**Team.28**
Achievement supported by 4.2, 4.3

**Communicate, align, and collaborate with multidisciplinary team members.** As described in Section 3.4, teams for AL projects, or more generally for AI projects, comprise members with diverse expertise and backgrounds. The best practice focused on the communication and collaboration of these teams is again significantly dependent on the team culture. For example, effective collaboration in these multidisciplinary teams involves exchanging ideas, knowledge, and the various perspectives inherent in different roles (Faraj *et al.* 2000).

Several aspects are conducive to fulfilling this best practice. These include shared communication channels, which can especially be implemented with appropriate tools. Other vital aspects are workshops, training sessions, an open team culture, and regular interdisciplinary meetings. Therefore, the proposed concepts do not per se fulfill this best practice but can provide significant added value. Suppose the technical team, which consists of different roles, follows the agile development approach, implements the infrastructure setup described in Figure 4.3, and works according to the proposed branching workflow. In this case, collaboration is centered around a central location for information exchange. While software engineers might work on planned changes to the application code in feature branches, ML-related tasks, such as those handled by a data scientist, can utilize experiment branches and the experiment tracking service to record their progress and communicate the results to the entire development team in a status meeting.

**Enforce Fairness and Privacy.** This best practice emphasizes the need to consider and enforce principles of fairness and data protection at all stages of the development and deployment of ML models. This includes implementing measures to prevent biases and avoid discrimination, as well as ensuring that the collection, processing, and use of (training-) data respect individuals' privacy and comply with applicable data protection policies.(Breck *et al.* 2017b; Google 2023b)

**Governance.29**

Fulfillment is not prevented.

The fulfillment of this best practice means that further steps must be taken, ranging from fairness analyses and tests to extensive data protection measures like data encryption and anonymization. We argue that the described concepts do not hinder the implementation of these measures. Instead, they contribute positively by promoting feedback mechanisms and continuous improvement, two essential aspects relevant in enforcing fairness and data protection.

**Table 4.1:** Aggregated representation of best practices, collected by Serban *et al.* 2020 and their satisfaction regarding the presented concepts: + + + matches a complete fulfillment. Scores ++ and + are intended to provide an evaluation of whether the proposed process methodology is an enabler for achievement. For the best practice marked with *o*, the proposed methodology is no enabler, but in our view the, compliance would not be hindered.

| Nr. | Title | Fulfillment |
|---|---|---|
| Data-1 | Use sanity checks for all external data sources | ++ |
| Data-2 | Check that input data is complete, balanced and well distributed | ++ |
| Data-3 | Write reusable scripts for data cleaning and merging | +++ |
| Data-4 | Ensure data labeling is performed in a strictly controlled process | ++ |
| Data-5 | Make datasets available on shared infrastructure (private or public) | ++ |
| Training-6 | Share a clearly defined training objective within the team | + |
| Training-7 | Capture the training objective in a metric that is easy to measure and underst. | + |
| Training-8 | Test all feature extraction code | ++ |
| Training-9 | Assign an owner to each feature and document its rationale | +++ |
| Training-10 | Actively remove or archive features that are not used | + |
| Training-11 | Peer review training scripts | +++ |
| Training-12 | Enable parallel training experiments | +++ |
| Training-13 | Automate hyper-parameter optimisation and model selection | ++ |
| Training-14 | Continuously measure model quality and performance | +++ |
| Training-15 | Share status and outcomes of experiments within the team | +++ |
| Training-16 | Use versioning for data, model, configurations and training scripts | +++ |
| Coding-17 | Run automated regression tests | ++ |
| Coding-18 | Use continuous integration | +++ |
| Coding-19 | Use static analysis to check code quality | ++ |
| Deployment-20 | Automate model deployment | +++ |
| Deployment-21 | Continuously monitor the behaviour of deployed models | ++ |
| Deployment-22 | Enable shadow deployment | ++ |
| Deployment-23 | Perform checks to detect skews between models | ++ |
| Deployment-24 | Enable automatic roll backs for production models | +++ |
| Deployment-25 | Log production predictions with the model's version and input data | ++ |
| Team-26 | Use a collaborative development platform | ++ |
| Team-27 | Work against a shared backlog | ++ |
| Team-28 | Communicate, align, and collaborate with multidisciplinary team members | ++ |
| Governance-29 | Enforce fairness and privacy | *o* |

**Summary.** To enable a final evaluation of the process methodology based on best practices from the literature, the results are presented in aggregated form in Table 4.1.

We have reviewed (a) which of the available aspects are fully met by the presented concepts, (b) whether additional measures are required to satisfy established best practices, or (c) are not feasible. As indicated in Table 4.1, we consider

- (a) 10 out of 29 best practices as inherently fully met. Most of these best practices fall within the category of Training.
- (b) 18 of the 29 best practices are not per se fulfilled by implementing the concepts. However, the proposed development methodology can be a facilitator for their implementation. While some of the best practices, such as Training-8, Deployment-22, or Deployment-23, can be met by extending the concepts, further steps in the development process, for example, through suitable tooling, are necessary for best practices Training-6, Training-7, and Training-10.
- (c) The proposed concepts impede none of the 29 best practices in their implementation. In other words, Teams following the development process are not prevented from adhering to the specific best practices, although additional methodological steps are required to achieve full compliance.

### 4.5.2 Expert Interviews

In MLOps research, significant importance is attributed to the industry's perspective and the academic world. As demonstrated by Giray 2021, expert interviews have been utilized to find answers to practice-oriented questions in the field of SE for ML projects. This method has been employed in other studies within this research domain, such as by Kreuzberger *et al.* 2023 and Steidl *et al.* 2023. Inspired by this approach, which underscores the relevance of industry feedback in this discipline, we decided to integrate this tactic into evaluating the proposed concepts.

Our approach includes the involvement of industry experts to discuss and evaluate the development methodology for data-centric AI projects that we propose. For this purpose, professionals from various industries and company sizes were selected, as shown in Table 4.2, to cover as diverse a perspective as possible with different industrial requirements and experiences. The eight participants held different technical roles and, in some cases, leadership positions. The company sizes ranged from Small and Medium Entrepreneurs (SMEs) with fewer than 500 employees to large corporations with more than 100,000 employees.

**Table 4.2:** Profiles of interviewees

| No. | Role/Position | Industry (Employees) |
|---|---|---|
| $\alpha$ | SE Team Lead | Technology (> 100k) |
| $\beta$ | SE Quality Manager | Technology (< 500) |
| $\gamma$ | AI Project Manager | R&D (≈ 30k) |
| $\delta$ | Data Scientist | R&D (≈ 2k) |
| $\epsilon$ | Head of AI | Consulting (< 500) |
| $\zeta$ | ML Engineer | Consulting (< 500) |
| $\eta$ | Data Scientist | Automotive (> 100k) |
| $\theta$ | Head of AI | Automotive (> 100k) |

### Interview procedure

Prior to the interviews, participants were sent preparatory materials.[5] These consisted of slides that explained the key concepts and the proposed workflow. The goal was to provide participants with a foundation for understanding and to enable a critical assessment of the proposed method.

The interviews themselves were conducted online as 60-minute face-to-face conversations. Here, the semi-structured interview technique facilitated a free-flowing yet content-directed conversation.

The conversation was structured in three phases:

- (1) The first phase, which was intended to last about 25 minutes, served to create the profile of the interview participant. They were asked about their current team role, their position, and their job activities. In addition, they were questioned about their experience in AI projects and the scale of these projects.
- (2) In the subsequent phase, which took about 10 minutes, the method was presented, and open questions regarding the principles of code, data, and automation, as well as the branch-based workflow, were clarified. This step was to ensure that the participants understood these.
- (3) In the third phase of the interview, which was planned to last 25 minutes, the participant was asked to evaluate the introduced concepts. As inspiration, the following three guiding questions were provided in the interview materials:
    - For which MLOps principles do you see a technical debt enabler through the presented development approach?
    - How do you evaluate the proposed development methodology according to process requirements?
    - Where do you see the biggest gaps and opportunities for you and your team?

We used the data collected from the conversations for a qualitative analysis, which will be discussed later. First, we will address some observations that emerged from the first phase of the interviews.

### Inventory of AI project maturity levels

In the traditional SE context, maturity models often play a vital role in the categorization of projects. These models provide a framework to measure a project's progress and maturity level across various degrees, which, in turn, allows for analysis and comparability. For this purpose, different maturity levels for ML projects were initially collected and classified. The maturity models from Microsoft Corporation 2021, Google 2023a, and John *et al.* 2021 were consulted and organized into the structure proposed by Liker 2004 along the dimensions of people, process, and technology.

Figure 4.8 provides an overview of the different aspects of AI project maturity levels in a triangle where each axis represents a different dimension according to Liker 2004. The color-coded levels represent the various developmental stages of AI projects. Level 1 corresponds to an initial project phase where most work is done manually, progressing to

[5] The materials are available under `https://github.com/ds-lab/slides`

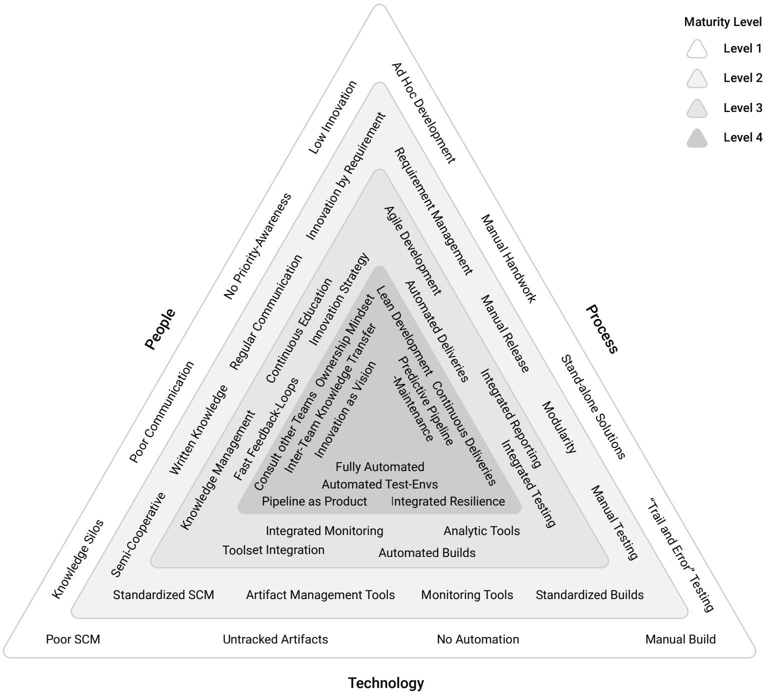


**Figure 4.8:** Project maturity levels along the dimensions of people, process and technology: Aspects based on AI/ML project maturity models from Microsoft Corporation 2021, Google 2023a, and John *et al.* 2021 were, according to Liker 2004, divided into three dimensions: people, process and technology. The levels increase from level 1 (= early project stage, most manual) to level 4 (= highest mature, fully automated).

level 4, which corresponds to a fully matured project with aspects of a fully developed stage.

In the first phase of the interview, participants were asked to provide a self-assessment of their organization and the projects they had conducted. Table 4.3 presents the results of this survey, with the assessment reflecting a subjective perception and not based on strict criteria. Nevertheless, this information is relevant to gaining insight into the status quo of the interview partners and taking their experience level into account when evaluating the methodology.

**Table 4.3:** Inventory of project maturity levels and self-assessment of interviewees. Interviewees were asked at the beginning of the interviews to use the table to rank the maturity of their projects.

| | **Technology** | **Process** | **People** |
|---|---|---|---|
| **Level 1** | Poor SCM $^{\gamma,\delta}$ | Ad Hoc Development $^{\delta}$ | Knowledge Silos |
| | Untracked Artifacts $^{\beta,\delta}$ | Manual Handwork $^{\gamma}$ | Poor Communication |
| | No Automation $^{\delta}$ | Stand-alone solutions $^{\gamma,\delta}$ | No Priority-Awareness |
| | Manual Build $^{\gamma,\delta,\epsilon}$ | „Trail and error“ | Low Innovation |
| **Level 2** | Standardized SCM $^{\epsilon}$ | Requirement Management $^{\epsilon}$ | Semi-Cooperative $^{\delta}$ |
| | Artifact Management Tools $^{\gamma}$ | Manual Release $^{\delta,\epsilon,\zeta}$ | Written Knowledge |
| | Monitoring Tools $^{\gamma,\epsilon,\zeta}$ | Modularity $^{\epsilon}$ | Regular Communication |
| | Standardized Builds | Manual Testing $^{\gamma,\delta,\epsilon}$ | Innovation by Requirement $^{\beta,\epsilon}$ |
| **Level 3** | Integrated Monitoring $^{\beta,\eta,\theta}$ | Agile Development $^{\beta,\zeta}$ | Knowledge Management $^{\beta,\gamma}$ |
| | Toolset Integration $^{\zeta,\eta,\theta}$ | Autom. Deliveries | Fast Feedback-Loops $^{\alpha,\zeta}$ |
| | Analytic Tools $^{\beta,\eta,\theta}$ | Integrated Reporting $^{\beta,\zeta,\eta}$ | Continuous Education $^{\beta}$ |
| | Autom. Builds $^{\zeta}$ | Integrated Testing $^{\beta,\zeta,\eta}$ | Innovation Strategy $^{\delta}$ |
| **Level 4** | Pipeline as product $^{\alpha,\zeta}$ | Lean Development $^{\alpha,\eta,\theta}$ | Inter-Team Transfer $^{\alpha,\epsilon,\zeta,\eta,\theta}$ |
| | Fully Automated $^{\alpha}$ | Continuous Deliveries $^{\alpha,\beta,\eta,\zeta}$ | Consult other Teams $^{\beta,\gamma,\delta,\epsilon}$ |
| | Integrated Resilience $^{\alpha}$ | Predictive Pipeline | Ownership Mindset $^{\gamma,\delta,\epsilon,\zeta,\eta,\theta}$ |
| | Autom. Test-Envs $^{\alpha,\beta,\eta}$ | - Maintenance $^{\alpha}$ | Innovation as Vision $^{\alpha,\zeta,\eta,\theta}$ |

Notably, interviewees from the R&D sector concurred on data science-driven processes, and technological aspects were aligned with a lower maturity level. For example, a Data Scientist ($\delta$) and an AI Project Manager ($\gamma$) reported 'poor SCM' and 'manual build', indicating initial challenges in software configuration management and build processes. These initial challenges were mirrored in the process domain through 'ad-hoc development' and 'stand-alone solutions', reported by these R&D sector interviewees, pointing to a lack of structured development processes.

This observation aligns with the study of Serban *et al.* 2020, who found in a quantitative survey of $n = 350$ respondents that adopting their identified practices, which were discussed in Section 4.5.1, was most prevalent in tech companies. At the same time, the organization type *„Research"* exhibited the lowest implementation rate.(Serban *et al.* 2020)

At level 2, the introduction of 'standardized SCM' and 'artifact management tools' by the Head of AI in consulting ($\epsilon$) and the AI Project Manager ($\gamma$) reflects a move towards more regulated and tool-supported environments. Similarly, the SE Quality Manager in technology ($\beta$) and the Head of AI ($\epsilon$) noted improvements in 'requirements management' and 'modularity' within the process domain, suggesting an evolution towards more refined development practices.

At level 3, approaches become apparent with 'integrated monitoring' and 'automated builds', specified by Data Scientists in the automotive industry ($\eta$ and $\theta$) and the ML Engineer ($\zeta$). 'Agile development' and 'integrated testing' were indicated at this level by the SE Quality Manager ($\beta$) and the ML Engineer ($\zeta$), indicating maturity in adaptive and continuous integration practices.

The highest maturity level 4 reveals a highly advanced state, where 'pipeline as product' and 'fully automated test environments' are indicated by the SE Team Lead in technology ($\alpha$) and the Head of AI in automotive ($\theta$), pointing to a fully matured and automated project environment. In the people dimension, 'ownership mindset' and 'innovation as vision' are attributed to various roles, including the Head of AI in consulting ($\epsilon$) and the Data Scientists in automotive ($\eta$ and $\theta$).

The distribution of the eight interviewees appears balanced across technology- and process-related implementations. However, a significant portion attributes a higher maturity level to person-related factors. This investigation, limited to a small group of respondents, suggests that methodological issues in developing AI projects at the time of the survey predominantly prevail in process- and technology-related approaches rather than human-centric work practices.

### Quantitative analysis of the assessment through expert interviews

As mentioned in the interview procedures, following the presentation of the proposed methodology and the clarification of questions by the interviewees, the participants were asked to assess the applicability in their practical work during the third phase of the interview. Here, stimulated by the three questions provided, they were to address aspects concerning people, process, and technology in their feedback.

**People.** In the analysis of the interviews, statements from participants focusing on the aspects of the people dimension revealed a crucial importance of team dynamics and collaboration in the context of AI project development methods.

Firstly, candidate $\epsilon$ underscored the significance of exchange within the team, emphasizing that solitary work on projects is not an option. They highlighted the necessity of communication within the team and with clients to develop a deeper understanding of domain knowledge. These considerations illustrate the need to foster an environment where knowledge exchange is paramount, aligning with the maturity levels of a collaborative culture where knowledge transfer and shared visions play a key role.

Moreover, $\epsilon$ viewed the proposed methods and processes as a solid foundation for discussion within the team, to be tailored to specific needs and team dynamics. They conceded that only certain concepts of the proposed methodology are relevant for most of their projects, suggesting that team collaboration could be enhanced by adopting and adapting concepts necessary for a specific project. $\gamma$ concurred, emphasizing the general need to select methods according to team size and pointing out the importance of scalability and flexibility of processes to suit different team structures and project scopes.

$\zeta$ brought forth an argument related to the developer dataset, advocating that the approach can simplify the developer's work as it obviates the need to manage the complexity of larger datasets. They proposed that significant data changes ideally trigger an automatic update of this developer dataset, creating an ergonomic environment for development while maintaining awareness of data dependencies. This approach underscores the necessity of practical, developer-oriented solutions that strike a balance between easy development and the underlying complexities of ML projects. This could lead to strengthened collaboration among heterogeneous roles, as the often very different working methods of data scientists and software developers could be harmonized.

**Process.** From the process-related perspective, candidate $\epsilon$ reflected the tendency for a discrepancy between established processes and those actually practiced. They stressed the need for robust processes that support both the ML pipeline and experimental aspects. $\epsilon$ advocates for pragmatic and straightforward approaches, which, from their experience, function well even in challenging tasks and high-pressure project situations. However, a weakness they noted in the shown concepts and workflows was the added complexity, especially concerning the proposed automation principles, a sentiment echoed by $\delta$.

In contrast, $\zeta$ highlighted the advantages of automation principles, suggesting they make the process less prone to errors and improve consistency. $\gamma$ spoke to the importance of structured development and testing processes for efficient and targeted work, especially in larger teams. They mentioned the benefit of a branching workflow that allows for parallel work with different data configurations and experiments, enabling the testing of various scenarios early in the project phase.

$\delta$ pointed out that standardized processes might be a disadvantage during exploratory project phases, such as feasibility studies, potentially limiting developers who prefer more individualized approaches. $\zeta$ concurred, noting that the outlined automation principle and consistent use of CI/CD/CT runners could lead to additional effort, which might necessitate infrastructure setup changes when unforeseen requirements arise. $\epsilon$ suggested that aside from experiment branches, a kind of laboratory environment for notebooks should be provided for data science-driven tasks.

On the topic of implementation for smaller projects with team sizes of fewer than five members, $\beta$, $\gamma$, $\delta$, $\epsilon$, and $\zeta$ expressed skepticism. They highlighted the considerable overhead in the initial phase as a critical concern, given that small AI projects are often still proof of concepts today.

As a project progresses, $\gamma$ and $\zeta$ acknowledged that the workflow with dedicated experiment branches offers the advantage of being transparent for the team while being isolated from the main development. $\gamma$ praised the concept of integrating automated testing routines into CI/CD/CT runners, which could be used for model validation steps. $\zeta$ commended the rapid feedback facilitated by quick iterations using a developer dataset optimized for running ML pipeline code swiftly.

$\alpha$ advanced the discussion on the data principle, proposing that differently sized development datasets could allow for an additional integration test, where multiple CI/CD/CT runners could be triggered in parallel by a commit, each checking out development datasets of varying sizes up to the full datasets. This approach compromises between rapid feedback for developers and testing on complete datasets.

Lastly, $\beta$, in their role as SE Quality Manager in consulting, emphasized, „Software quality is crucial. It starts with structured planning," endorsing the proposed concepts of code principles as fundamentally sensible.

**Technology.** Focusing on the aspects of technology, the interviewees provided feedback on technical approaches and the challenges of managing and versioning data in the context of data-centric AI projects.

Candidate $\epsilon$ recognized the presented technical approaches as relevant and crucial for achieving reproducible results, emphasizing the significance of tools that assist in data management and versioning, especially in dynamic scenarios such as AL projects with an evolving set of labels.

Both $\epsilon$ and $\zeta$ noted the importance of maintaining a clear overview of the different versions and experiments. They pointed out that as the complexity and number of experiments and versions increase, the challenge arises to establish a consistent reference between all elements efficiently and swiftly. In the context of scalability, $\alpha$ and $\beta$ concurred that for large projects, the multitude of experimental branches could be confusing when running in parallel with other branches.

$\gamma$, $\delta$, and $\zeta$ emphasized the necessity for the developer dataset to be as representative as possible to create realistic development conditions. $\gamma$ mentioned that while ML results do not depend on it, an inaccurate developer dataset could misdirect the project if the development team relies too heavily on it as a basis for developing ML pipeline code and other non-ML-related application code.

Hence, $\alpha$, $\beta$, $\gamma$, and $\epsilon$ believe that the proposal to use a developer dataset is fundamentally sound but stresses the importance of focusing on the requirements for selecting suitable instances. The general idea of providing minimal datasets for development on the developer client or within a developer container appears promising.

$\delta$ reported that the significant challenge in the technological implementation of concepts is the lack of expertise of the team members: „Usually, the technologies are available, but people are needed to set it up, maintain it, etc., and to support it.“

$\beta$ stated, „In AI projects, the major hurdle is to conduct data management and administration, particularly in terms of revision, security, and traceability,“ which would be significantly improved by artifact management arising from the proposed principles. $\beta$ further emphasized that AI projects are very specific and heavily dependent on the respective domain. The same applies to selecting technology decisions: „Technology is so fast-moving that focusing on one toolchain might be a dead end.“ However, the concepts presented were considered agnostic enough to allow using different tools.

**Summary.** Through the interviews of eight experts from the industry, it became evident that the proposed methodology establishes a solid foundation for developers of data-centric AI projects, particularly when regular feedback is envisaged, as in the case of integrating an AL-loop. The demonstrated principles on data, code, and automation address challenges concerning the resulting dynamics in artifacts while simultaneously increasing transparency and collaboration in teams composed of diverse technical roles.

The requirements mentioned for the development dataset offer potential for discussion. Moreover, it is observed that adopting all concepts, especially in terms of automation, introduces an inevitable overhead into a project, which can outweigh the benefits, particularly in the early exploratory phase.

Overall, the different perspectives collectively indicate that while the overarching methodology provides a structure, its application must be attuned to the specific facets of team interaction, the details of project requirements, and the practical aspects of workflow management. However, all interview participants agreed that teams benefit most from the provided principles by improving traceability.

### 4.5.3 Metadata Analysis

In extension to the evaluations from Section 4.5.1, where the engineering methodology proposal were analyzed for its alignment with established best practices, and the inclusion of industry experts through semi-structured interviews in Section 4.5.2, the third part focuses on a particularly practical evaluation perspective. For this purpose, the two projects, which are detailed in Chapter 6, were closely examined.

Within the utilized SCM application GitLab, data was collected over the course of the project's duration, which could be accessed using an API. This approach enabled the retrieval of metadata from repositories and the analysis of metrics from the CI/CD/CT runners. This is a common method to evaluate the duration of pipelines and their success rates over time (GitLab 2023).

The introduced branching workflow and the establishment of separate branch namespaces allowed for the segregation of pipeline statistics for different branches, followed by subsequent calculations. These results aim to assess the efficiency of the pursued developmental principles.

While the technical implementation of the skin image analysis use case, as well as of the ECG signal classification use case, which will be both detailed in Chapter 6, differ significantly in terms of the ML pipeline, and this must be kept in mind when comparing the two projects, there are further technical specifications that apply uniformly to both projects. In both projects, the CI/CD/CT runners were hosted on a GPU server. For data versioning, we integrated DVC (Iterative 2020) and MLflow (Zaharia *et al.* 2018) to track experimental tasks. The technical and organizational differences that impact the statistics of each use case are outlined below.

**Project-specific Characteristics, affected the Implementation of the Skin Image Analysis Use Case**

Two GPUs were available for model training in the project where the ML pipeline for skin image analysis was implemented. These GPUs could process various tasks in parallel and were exclusively available to the team during the implementation phase. Data management, including the storage of raw data and all other computed artifacts, such as preprocessed data and trained models, was realized with a file server and SSH access.

A total of 16 developers contributed to this project. The experienced core team consisted of five developers in various roles, applying the described development method for the first time. We determined the size of the development dataset to be 100 samples. This quantity was selected to be large enough to encompass instances of each class for training-test splits yet small enough to execute the ML pipeline code on the developers' client computers within a reasonable timeframe.

The analysis of the CI/CD/CT pipelines, as depicted in Figure 4.9, reveals that over a 12-month period, a total of 1,850 jobs were executed, with the majority triggered in feature branches or the main branch. As illustrated in the right part of the figure, using the development dataset proved advantageous.

Developers received feedback on whether the pipeline had failed in an average time of 6.36 minutes. In cases of successful pipeline execution, results were available within an average of 18.58 minutes. Notably, this marks a significant time-saving compared to the average execution time of experiment branches, which was 179.87 minutes (factor 28) for failed executions and 391.11 minutes (factor 21) for successful ones.

Considering the substantially higher number of pipeline executions on code-focused feature- and main branches, the concepts emphasize the

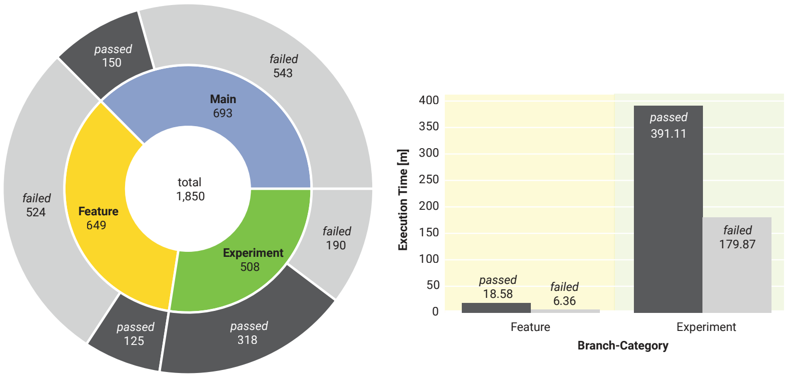


**Figure 4.9:** Overview of runner execution metadata in the skin image analysis use case project. Within the CI pipelines, a total of 1,850 jobs were executed, primarily in feature- and main branches. The average execution time of the jobs (right bar chart) indicates a significant difference between feature- and experiment branches, with developers benefiting from fast feedback during development iterations.

promotion of agility for the development team, especially in optimizing time and computational capacities.

### Project-specific Characteristics, affected the Implementation of the ECG Signal Classification Use Case

In the project implementing the ML pipeline for ECG signal classification, we adapted the infrastructure to meet the increased complexity and requirements of the use case. This time, four GPUs were exclusively available to the team, capable of parallel processing various tasks. An S3 bucket was used as the data remote, storing and exchanging raw data, preprocessed data, trained models, and other computed artifacts.

A total of 19 developers contributed to this project. Following the identical engineering methodology proposal, the core team remained unchanged from the skin image analysis project, thus capitalizing on initial learning experiences. In this project, the development dataset again comprised 100 samples assembled from various data sources to meet the specifications outlined in Section 4.2.1.

Figure 4.10 presents the analysis of the CI/CD/CT pipelines over an 18-month observation period. A total of 3,048 jobs were triggered, including automated nightly retraining of the model on the production branch. For this project, we recorded the job execution time on the main branch, intended for integration tests, highlighting resource savings from using development datasets.

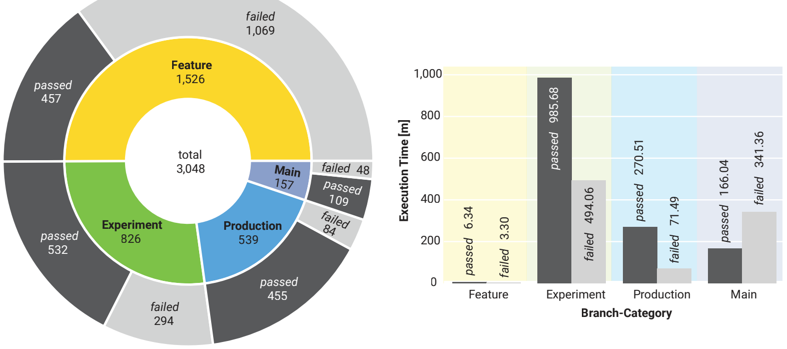


**Figure 4.10:** Overview of runner execution metadata in the ECG signal classifier use case project. In this project, 3,048 CI pipeline jobs were executed, including automated nightly retraining of the model in the production environment. The feedback cycles for developers were significantly shortened through the implemented concepts, as evidenced by the average job execution times shown in the right bar chart.

Despite runners checking out entire datasets in both the production and experiment branches, a clear difference in execution time was observed. This discrepancy can be attributed to the use of experiments for more complex simulations and the fact that the datasets in the production environment did not change daily, allowing the deployment of the already trained model without modification.

Significant time savings were evident in the feedback cycles. In feature branch implementations, the average execution duration of failed jobs was 3.30 minutes – a factor of 149 compared to experiments. For successful jobs, the average duration was 6.34 minutes, a factor of 156 compared to experiment branches.

This project is another example of successfully adapting the development methodology to an data-centric AI project, demonstrating how the principles can lead to considerable efficiency and productivity gains.

Using the principles and development workflow, both time and computational resources were saved while enhancing the agility of the development team through rapid feedback cycles. The concepts proved beneficial for project quality assurance: Integration tests for the ML pipeline within a CI/CD/CT job on the main branch provided feedback to the team within an average execution time of about 166 minutes for passing the test stage, with the ML pipeline running using the development dataset and various hyperparameter combinations.

### Conclusions from the Metadata Analysis

Adopting the introduced engineering methodology in both projects proved advantageous for agility and team collaboration. Due to the project and code structure, multiple developers could simultaneously implement new features and initiate experiments. Including the development dataset and using different runners led to significant differences in the execution times of the CI/CD/CT pipelines, enabling rapid iteration during code-focused implementation and considerable resource savings throughout the projects.

However, in the practical application of these principles, some aspects cannot be overlooked. Firstly, there are substantial requirements for the development dataset, as its creation can be challenging due to often very heterogeneous data sources. Furthermore, it may become necessary to maintain different development datasets of various sizes.

Furthermore, the other principles, such as using runners for conducting experiments or practicing modular coding, represent an inevitable overhead for team members who need to become more accustomed to this way of working. The team requires a member capable of setting up the necessary infrastructure, and every team member must be familiar with both code and data version control systems. Developers or data scientists unfamiliar with a data version control system will face a steep learning curve and frequent pitfalls, such as overly large merge requests. Feature branches will likely require a different configuration than the best model, for example, fewer training epochs to facilitate the rapid feedback loop. These different configurations necessitate additional management.

In addition to enhancing traceability, which was confirmed in the evaluation through expert interviews, the teams can benefit from rapid feedback cycles once the concepts have been implemented. Transparent collaboration is enabled for members of various technical roles, which, combined with established best practices from agile software development, leads to positive effects in the project.

## 4.6 Summary

This chapter introduces an approach for developing data-centric AI systems. Guidelines have been presented to assist engineering teams in effectively structuring a project, especially when it progresses from a feasibility study to a mature one. A focus is placed on harmonizing the collaboration of different team members as well as managing data and code versioning to enhance traceability and enable teams to track resulting artifacts, such as trained models, throughout the entire life cycle.

To this end, three principles were introduced which concern the handling of data, the code and project structure, and in terms of automation, a minimal infrastructure and the use of runners for CI/CD/CT. Additionally, the development methodology proposes a branch-based workflow that introduces different namespaces for branches which fulfills functions with a stronger focus on either data or code.

Finally, the concepts are evaluated through a three-stage process, initially discussing the fulfillment of best practices from the scientific literature in detail, followed by assessing practical suitability through expert interviews. A metadata analysis provides a technical perspective, evaluating two projects in which the proposed development practices were implemented.

The presented methodology approach, including its principles, combined with the branching workflow, serves as a process guide for engineering teams, which can be used independently of the tools employed to fulfill requirements in developing trustworthy AI systems.

# III. Design of AL Projects
# A System Perspective

# Framework Design Approach 5

In the implementation of AL projects, the importance of a structured framework increases as concepts of trustworthy AI, such as transparency and traceability, become relevant. The dynamic nature of artifacts, driven by continuous learning and evolving datasets, necessitates implementing techniques for tracking data provenance, versioning code, data, and models and applying methods for model interpretation and stakeholder-specific UIs. Traceability in the sense of consistent versioning of all artifacts is a crucial MLOps principle that significantly contributes to the compliance and trustworthiness of an AI system (Kreuzberger *et al.* 2023).

However, currently available AL frameworks often lack the integration of concepts that increase transparency and traceability, which become necessary for a comprehensive understanding and replication of the learning process from a systemic perspective. This Chapter introduces LIFEDATA, an innovative framework for traceable AL projects, developed as part of a BMBF project[1] of the same name.

Alongside a conceptual description, the framework comprises open-source software consisting of a core framework and a project template, aiming to enhance the transparency in AL projects.

LIFEDATA presents a novel approach that enables researchers and practitioners to comprehend the learning process, from training sample selection to an interpretable ML model output, thereby improving the overall understanding of the ML model's behavior. The following Chapter will present the technical design of LIFEDATA and provide a structured approach to AL projects.

In addition to the concepts first presented in the publication Stieler *et al.* 2023b, the associated open-source software, which is implemented in Python, essentially consists of two elements that are published under the GitHub repositories:

- `https://github.com/ds-lab/lifedata`
  The core framework with various interfaces, a database, and a web-based UI for annotations.
- `https://github.com/ds-lab/lifedata-project-template`
  The project template, which LIFEDATA users can use as a starting point for creating an AL project. It provides boilerplate code for data scientists and enables the initialization of AL project instances with the required framework structures.

**5.1 Related Work . . . . . . 124**
**5.2 Overall Process . . . . . 127**
**5.3 Core Framework . . . . 128**
5.3.1 User Management and Interfaces . . . . . . . . . 128
5.3.2 Annotation Domain Layer 131
5.3.3 Data Persistence Layer . 133
5.3.4 Project Persistence Layer 136
**5.4 Project Instance . . . . . 137**
5.4.1 Basic MLOps Components . . . . . . . . . . . . 137
5.4.2 Data Import . . . . . . . 138
5.4.3 Machine Learning Service 139
5.4.4 Webinterface Components . . . . . . . . . . . . 143
**5.5 Limitations . . . . . . . . 144**
**5.6 Summary . . . . . . . . . 146**

[1] Reference number 031L9196B

## 5.1 Related Work

One of the first AL frameworks implemented in Python, introduced in 2017, is *Libact*. *Libact* is distinguished by its modularized design, which allows for the integration and customization of one's own QSs and ML models implemented in Scikit-Learn through uniform interfaces. In addition to integrating popular QSs, it supports the Active-Learning-by-Learning (ALBL) meta-algorithm introduced by Hsu *et al.* 2015, which validates different QSs in real-time with the goal of selecting the most effective strategy.(Yang *et al.* 2017)

*ModAL*, introduced in 2018, is another modular AL framework. In addition to its flexible and expandable design, *ModAL* allows for rapid prototype development and easy customization through its out-of-the-box implementation of various QSs, qualifying *ModAL* for simulations and integration in real AL projects. *ModAL* is characterized by its straightforward, object-oriented design and is fully compatible with scikit-learn models and workflows.(Danka *et al.* 2018)

*ALiPy*, released in 2019, is a Python-implemented toolbox for AL. The framework includes 20 QSs as well as its own approaches, like "AL with Noisy Oracles," demonstrating a focus on simulations. *ALiPy* is modular like *Libact* and *ModAL* and independent of model type, but is primarily used for conducting experiments. To this end, the framework supports various settings and includes methods for evaluating results. Its AL experiment module executes a classical AL loop with fixed batch-size and bounded cost, resuming failed experiments using checkpoints and allowing computation of various performance metrics.(Tang *et al.* 2019)

While *Libact*, *ModAL*, and *ALiPy* exclusively implement the logic around the AL loop, the *SMART* framework, published by Chew *et al.* 2019, covers the aspect of data annotation. In addition to user prompts for one or more human oracles, *SMART* provides an administrative dashboard with visualizations and metrics, which enable evaluations such as inter-rater reliability.

The *BaaL* framework, introduced in 2020, integrates Bayesian methods and addresses common real-world problems such as annotation errors and dataset imbalance. The query techniques specifically implemented in *BaaL*, like partial uncertainty sampling, allow for a data-centric view of model uncertainty, which is particularly relevant for AL projects with DL models.(Atighehchian *et al.* 2020)

*DeepAL*, by Huang 2021, is a Python framework for AL that focuses explicitly on DL models. Like *Libact*, *ModAL*, and *ALiPy*, its architecture is modular, but *DeepAL* can only integrate ML models implemented in PyTorch (Paszke *et al.* 2019). Furthermore, it aims to test various pool-based AL scenarios and supports a range of QSs similar to *BaaL*, including Bayesian methods.

Introduced in 2022, *cardinal* stands out for its metrics focus and addresses the resource intensity of AL simulations. It aims to provide insights through metrics rather than merely improving QSs, addressing implementation needs that other packages may not cover. This modular framework assists data scientists in selecting appropriate QSs for their use cases by efficiently calculating and logging metrics through cached experiments.(Abraham *et al.* 2022)

The research community has already introduced various open-source frameworks in the AL landscape. These provide valuable tools for implementing AL projects in Python and simulating learning processes to investigate the efficiency and effectiveness of model training. In addition to tools specialized in simulations, others focus on providing an extensive interface for data annotation.

Despite their similarities, such as the support of various QSs and flexibility through their modular architecture, these frameworks differ in their specific functions, supported scenarios, and design philosophies, as indicated in the comparison shown in Table 5.1. While the three most popular frameworks *ModAL*, *Libact*, and *ALiPy*, aim for broader and more flexible applications, other frameworks, like *DeepAL* and *BaaL*, offer specialized solutions for certain types of ML models. *SMART* is the only Python-implemented AL tool to provide an annotation interface, while *Cardinal* offers features to streamline resource-intensive AL experiments.

By integrating traceability and interaction concepts, LIFEDATA differs from the currently available AL frameworks. It includes versioning technologies and an interface for annotations, taking into account the origin of the labels. Before focusing on the system perspective, the following section provides a process-oriented depiction of the workflow for AL projects that utilize the LIFEDATA framework.

**Table 5.1:** AL framework comparison.*All frameworks provide differently implemented sampling algorithms and usually can be extended with custom QSs. Regarding the category, 'other' refers to the integration of proposals by the framework authors, primarily meta-based strategies. ***Cardinal* implements a caching logic for temporary storage of artifacts and metrics.

| | **Libact** | **ModAL** | **ALiPy** | **SMART** | **BaaL** | **DeepAL** | **Cardinal** | **LIFEDATA** |
|---|---|---|---|---|---|---|---|---|
| Reference | Hsu *et al.* 2015 | Danka *et al.* 2018 | Tang *et al.* 2019 | Chew *et al.* 2019 | Atighehchian *et al.* 2020 | Huang 2021 | Abraham *et al.* 2022 | Stieler *et al.* 2023b |
| Open Source Repository | `https://github.com/ntucllab/libact` | `https://github.com/modAL-python/modAL` | `https://github.com/NUAA-AL/ALiPy` | `https://github.com/RTIInternational/SMART` | `https://github.com/baal-org/baal` | `https://github.com/ej0cl6/deep-active-learning` | `https://github.com/dataiku-research/cardinal` | `https://github.com/ds-lab/lifedata` |
| Python Version | 3.6 | 3.5 | 3.4 | 3.8 | 3.8 | 3.8 | 3.5 | 3.10 |
| Query Scenario | Pool | Pool, Stream | Pool | Pool | Pool | Pool | Pool | Pool |
| Labeling Oracle | Simulation | Simulation | Simulation | Human | Simulation | Simulation | Simulation | Simulation, Human |
| Semi-Superv. Learning | no | no | no | no | yes | no | no | yes |
| QSs* | random, other, informative, representative | random, other, informative, representative | random, other, informative, representative | informative | random, informative, representative, hybrid | random, other, informative, representative | random, informative, representative | random, informative, representative, hybrid |
| ML Frameworks | Scikit-Learn, | Scikit-Learn | Scikit-Learn, TF | Scikit-Learn | PyTorch | PyTorch | Scikit-Learn | Scikit-Learn, TF, PyTorch |
| Annotation GUI | no | no | no | Web-based | no | no | no | Web-based |
| Label Provenance | ./. | ./. | ./. | yes | ./. | ./. | ./. | yes |
| ML Pipeline Orchestration | no | no | no | no | no | no | no | yes |
| Artifact Tracking | no | no | no | Labels, Model | no | no | no** | Code, Data, Labels, Model |
| Metrics Tracking | no | no | no | yes | no | no | no** | yes |

## 5.2 Overall Process

Central components and key actors are discussed in Figure 5.1, representing an integration of process-oriented and system-oriented viewpoints. It illustrates four main areas: Machine Learning, Labeling, Monitoring, and Exploitation, as well as three phases: Inception, Active & Semi-Supervised Learning, and Termination Assessment.

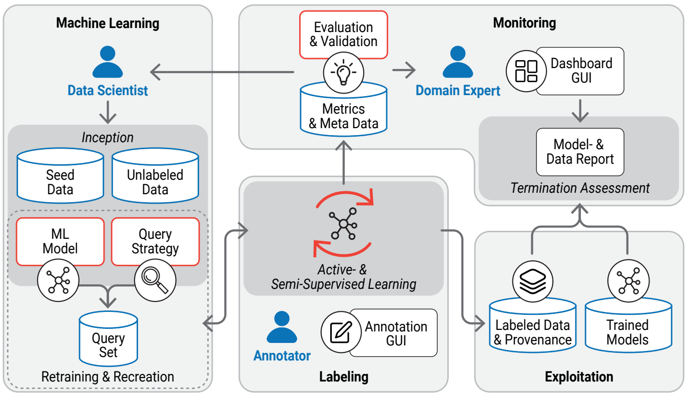


**Figure 5.1:** Process outline in an active learning project with LIFEDATA.

The Inception phase marks the starting point in Machine Learning and is positioned on the left side of the figure. Here, data scientists lay the groundwork for a pool-based scenario by assembling a corpus of unlabeled data. This can be supplemented with seed data, if available. As detailed in Section 3.2, this dataset consists of pre-labeled samples that can provide a foundational basis for training the ML model.

Further included in the Inception phase of Figure 5.1 are the components „ML Model“ and „Query Strategy“, which together form the framework for „Retraining & Recreation“ . These two elements lead to creating the queryset, which comprises the samples to be labeled in the current iteration. The bidirectional arrow connecting the Machine Learning area with the Labeling phase illustrates the provisioning of the Query Set for the labeling phase. In this phase, the acquisition of annotations occurs, provided either by human annotators via the Annotation Graphical User Interface (GUI) or automatically as pseudo labels in a semi-supervised learning scenario.

These newly generated labels are then recursively utilized for the retraining of the model in the subsequent AL cycle. In parallel, there is a pathway to the Monitoring area, where the effects of each training iteration are analyzed. This central area of the framework is responsible for collecting metrics and metadata. This information is made available to data scientists and domain experts to inform them about the state and performance of the model, as well as the progress of data annotation. Data scientists use the metrics and metadata to fine-tune the model and the QS, while domain experts have access to this information through a Dashboard GUI. The Dashboard includes results of the 'Evaluation & Validation' component, which encompasses methods for model interpretation and thus forms a crucial building block for promoting transparency and therfore trustworthiness.

The labeled data and their provenance, along with the trained models, are the essential components of the Exploitation phase, located on the right side of the figure. They, along with the evaluation and validation information, feed into the final assessment, which contains vital artifacts such as model and data reports and summarizes the project outcomes.

## 5.3 Core Framework

In the following section, the process view will be transformed into an reference architecture, which is displayed in Figure 5.2 and depicts its two key elements, the core framework, and the project instance.

Essentially, the LIFEDATA core framework is built in four layers and provides two interfaces, which for ease of overview, are colored green. Beginning with the top layer, a detailed introduction to each component is presented, elucidating their functionalities and interrelationships within the overall system.

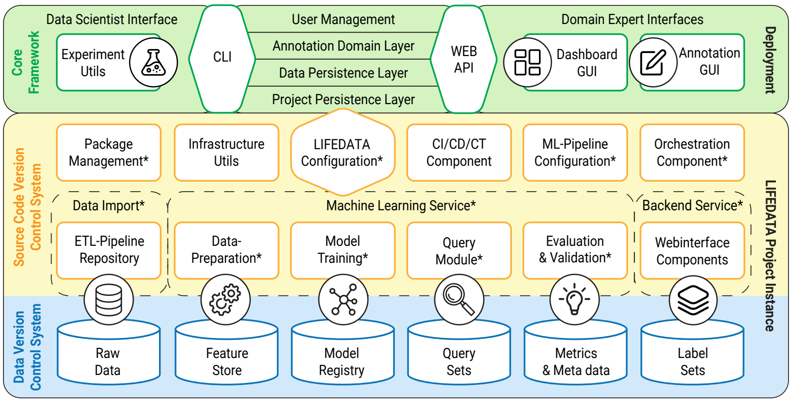


**Figure 5.2:** Reference architecture of active learning projects using LIFEDATA. The structure of the core framework (green) is displayed above the project instance. All code modules and configuration files (yellow) are versioned by a source code version control system, while the linked artifacts (blue) are managed by a data version control system. * = Component is part of the provided project template.

### 5.3.1 User Management and Interfaces

AL projects typically involve several actors with diverse technical skills. Section 3.4 categorizes these actors along the developmet life cycle into technical and non-technical roles, including data scientists, annotators and domain experts as presented in Figure 5.1.

As the definitions of these roles suggest, each actor within the project requires access to various functionalities and data. To meet the different requirements of these stakeholder, the LIFEDATA core framework has implemented both a Command Line Interface (CLI) as well as a WEB-API, which are subsequently described in more detail.

#### Command Line Interface (CLI)

The CLI serves as the primary entry point for stakeholders in technical roles and offers a range of commands for controlling the project instance. It has been implemented using the Python package Click, known as the „Command Line Interface Creation Kit“ (The Pallets Projects 2014), and enables interaction with various functions of the core framework, an overview, which is provided in Table 5.2.

**Table 5.2:** Overview of available commands integrated in LIFEDATA's CLI.

| Command | Option | Description |
|---|---|---|
| init | | Creates a project instance using the LIFEDATA project template.<br>Expected argument: project_name |
| start | | Main function that performs a full startup of all LIFEDATA services.<br>Optional argument: dev_option: activates development functions, which provides extended exception handling and options to reload services. |
| | backend | Starts the backend service. |
| | annotationsui | Starts the annotation GUI service. |
| | db | Starts the database service. |
| load | | Provides functions for loading samples into the database and for importing a queryset. |
| | samples | Creates entries for training samples in the database to make them available for annotation. |
| | queryset | Load entries of the produced queryset into the database. |
| dump | | Enables the export of data for data backup and transfer to the project instance. |
| | labelstate | Exports the labeled data as JSON file. |
| analyse | | Provides functions for basic statistical analysis related to the annotations and trivial data manipulations. |
| | label_frequency | Queries the frequency of the labels. |
| | label_cooccurrence | Creates a heat map showing the coefficients of coexistence of the labels. |
| | label_correlations | Calculates the label correlations. |

The use of the CLI is immediately available to the user after installing the lifedata Python package. To illustrate this more concretely, this section describes an excerpt of the commands that can be executed via the CLI and correspond to a typical project sequence, as shown in Table 5.2. Starting with the command

```
lifedata init <project_name>
```

the project instance detailed described in Section 5.4 can be created using the project template. Listing 5.1 displays a segment of its implementation. The function is decorated with the `@click.argument("project_name")` decorator, indicating that the CLI command expects a project name specified by the user as an argument. Using Cookiecutter (Greenfeld 2013), a cross-platform command-line utility, a project repository is created at the path specified in line 16.

Once the project instance is initialized, the services provided by the framework can be started using the CLI command `start`. Invoking the main command begins with the backend, the database, and the Annotation GUI, launching all necessary services. However, it is possible to specify which service to start with an option - a feature that becomes useful for distributing services in later deployment. For example, running the command

```
lifedata start db
```

will start the provided database described in the Data Persistence Layer. As seen in Table 5.2, functions are available within the `load` command group that creates entries of the samples to be annotated in the database.

**Listing 5.1:** Code snippet of the CLI logic for initializing a project using the project template.

```python
# /lifedata/cli/init.py
from pathlib import Path
import cookiecutter

# ...
@click.argument("project_name")
def init(
   project_name,
   # ...
):
   """
   Setup an AL project using LIFEDATA's project template
   """
   path = Path.cwd() / project_name.lower()
   print(f"Creating project in {path} ...")

   cookiecutter(
      "https://github.com/ds-lab/lifedata-project-template",
      no_input=True,
      extra_context={
            "project_name": project_name.lower(),
      },
   )
   # ...
```

On the other hand, the `dump` command group includes logic that can be used to export information from the database to make it available in other components, such as the ML service of the project instance.

Additional commands include `analyse`, which performs basic evaluations through a predefined database query, allowing the CLI user a quick analysis, for example, of the number of given labels.

### Web-Application Programming Interface (WEB-API)

The WEB-API covers various functions regarding authentication, data processing, user interaction, and system integration. By defining different endpoints, various services and functionalities of LIFEDATA are accessible to external requests.

Authentication mechanisms in LIFEDATA are realized through integrating the Identity and Access Management tool Keycloak, which covers user management (Keycloak Authors 2023) Within the WEB-API, JSON Web Tokens (JWT) functions are implemented, which Keycloak provides as an Authentication Response to grant the user access to requested resources. The defined authentication pipeline extracts the JWT, decodes it, and validates the user identity and their permissions.

As shown in Figure 5.2, the WEB-API extends over the subsequent layers. Given that the logic implemented therein is to be made available to other components, there is a provider that offers these as a web service using Representational State Transfer (REST).

An overview is presented in Table 5.3, which further gives a brief description. For a better orientation, the table is organized into the four layers of the LIFEDATA core framework, highlighting the specific aspects addressed by each.

To further elucidate the implementation, we will continue with the next layer.

**Table 5.3:** Overview of REST-API methods by layer provided in LIFEDATA. This overview categorizes methods into LIFEDATA's core framework layers User Management, Annotation Persistence, and Data Persistence as well as Project Persistence, detailing their role in authentication, sample handling, and system configuration within the LIFEDATA core framework.

| Layer | Method | Description |
|---|---|---|
| User Mgm. | provide_auth_token_decoder | Retrieves a token decoder based on project settings, for the token-based authentication processes. |
| | provide_authentication_service | Configures an authentication service, reliant on functionalities like the database session, event dispatcher, and token decoder. |
| Annotation Domain | provide_annotator | Provides annotator information based on the authentication service. |
| | provide_sample | Provides the sample, dependent on a database session, annotator, event dispatcher, and sample data repository. |
| | provide_sample_repository | Initializes and provides a repository for samples within the project scope. |
| | provide_queryset_db_repository | Initializes a query set repository, enabling data querying mechanisms. |
| | provide_annotation_db_repository | Establishes a repository for managing database-stored annotations. |
| | provide_annotation_count | Calculates and provides the count of annotations based on annotator information and the annotations database repository. |
| | provide_label_metadata | Retrieves label metadata in accordance with the label configuration. |
| Data Persistence | provide_db | Initializes a database session, ensuring life time management. |
| | provide_load_sample_display_data | Helper function that takes a sample_id and returns the corresponding sample data from the annotation widget. |
| | provide_queued_samples | Provides queued samples, depending on dependencies such as annotator information and the annotations repository. |
| | provide_sample_db_repository | Sets up a repository for handling samples within the database. |
| | provide_db_sample_state | Determines and provides the state of a sample in the database, using dependencies on sample and annotation repositories. |
| | provide_domain_controller | Configures a domain controller, based on database session and event dispatcher. |
| | provide_event_dispatcher | Creates an event dispatcher, utilizing a database session dependency for event management. |
| Project P. | provide_project | Loads the project configuration, facilitating project-specific operations. |
| | provide_model_training_service | Establishes and provides a model training service, contingent on project configuration and annotation repository. |

## 5.3.2 Annotation Domain Layer

The LIFEDATA core framework includes an annotation domain layer placed between the user- and the data corpus. It serves as bridge between these two entities and allows for the modeling of associations between labels and samples. Thus, the Annotation Domain Layer provides an abstraction that enables the definition of structured and semantic relationships between information about annotators, labels, and samples.

**Data Classes.** A core concept of the Annotation Domain Layer, the so-called data classes, specifies the structure of the defined entities above. For example, the data class `Annotation` describes how annotations are represented, while the data class `Annotator` outlines the attributes

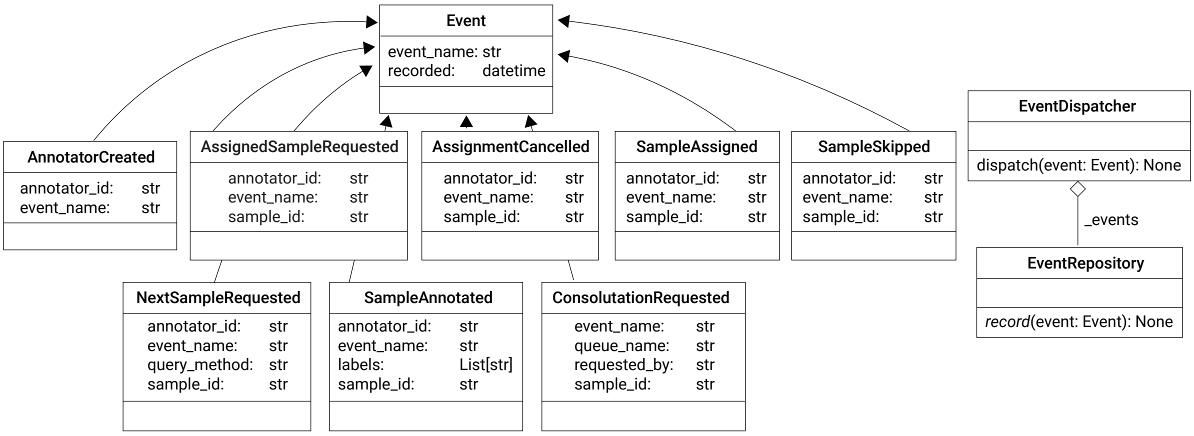


**Figure 5.3:** Class diagram of events in LIFEDATA. Each class represents an event that can occur during the annotation process, with the clas „Event“ serving as the base class with shared attributes „event_name“ as well as a timestamp. The „EventDispatcher“ manages the flow of these events, while the „EventRepository“ is for recording them.

associated with an annotator. Each data class is designed to capture all information relevant to the entity.

Listing 5.2 shows an example implementation of the data class `Assignment`, which includes the attributes `annotator_id`, `sample_id` and a timestamp set in `created`. It might be extended with additional attributes if required, however, this definition is sufficient for the example. Data classes can be defined using the `@dataclass`-decorator. As shown in the code snippet, they provide typing and the assignment of default values for fields during instantiation (line 11).

**Listing 5.2:** Defintion of the dataclass „Assignment“ in Python.

```
1  # lifedata/annotations/assignment.py
2  from dataclasses import dataclass
3  from datetime import datetime
4  from datetime import timezone
5  #...
6
7  @dataclass
8  class Assignment:
9      annotator_id: str
10     sample_id: str
11     created: datetime = field(default_factory=lambda:
       ↪  datetime.now(timezone.utc))
12
13     #...
```

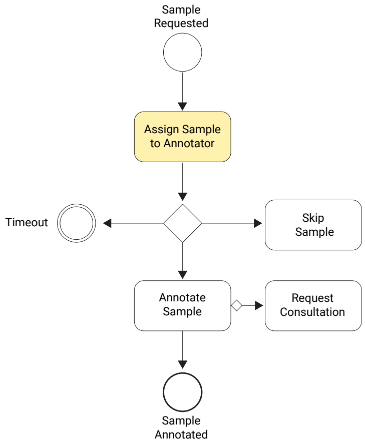
**Events.** An integral paradigm of LIFEDATA is ensuring traceability, which runs through the subsequent layers and is supported by event definitions in the annotation domain layer. This ensures that users' actions on the data corpus are traceable by recording what actions users have performed on the data while interacting with it, such as during the annotation phase. These event definitions are summarized in a class diagram, which is shown in Figure 5.3.

To better understand the significance of events and data classes, let's stay in the example and examine the implemented logic for the sample-annotator assignments in the Annotation Domain Layer. This functionality is crucial for managing the interaction between annotators and samples. It allows the assignment of new labels to samples and assigns samples to specific annotators.

The logic that implements the assignment of samples and annotators is defined in the `AssignmentService` class, a snippet of which is shown in Listing 5.3. As with the REST-API call `provide_sample` (cf. Table 5.3), the class dependencies `EventDispatcher`, `SampleProjectRepository`, and `AssignmentRepository` are initialized when creating an instance.

The method `get_sample(...)` returns a `sample_id` while ensuring that the sample is assigned to the annotator. To this end, it first checks existing assignments, ensuring an active assignment exists for the specified annotator and updates it if necessary. If no active assignment exists for the annotator, a new assignment is created, and corresponding events are triggered through the `event_dispatcher`.

Although Listing 5.3 does not show the complete implementation of the `AssignmentService`, the concept of events and data classes is illustrated in the `get_sample(...)` method. After the assignment is created in line 23, and added to the assignment repository in line 24, the `SampleAssigned` event is triggered in line 26 before returning a `sample_id`.



Simplified flowchart showing the course of a sample-annotator relationship in LIFEDATA.

The implementation of the logic for assigning annotations follows the same scheme, using the defined data class for Annotation and the corresponding events. Another aspect of the Annotation Domain Layer is handling sample skipping and additional label requests. This functionality is crucial to support flexible annotation scenarios, such as AL with multiple views as described by Muslea *et al.* 2006, or to implement mechanisms for multiple annotations of specific samples by different annotators to investigate inter- and intra-annotator agreement.

Regardless of the annotation scenario, the definition of events ensures the recording of annotator interactions and thus establishes the data provenance of the annotations to ensure transparency and traceability. This information primarily converges in the next layer, which will be examined subsequently.

### 5.3.3 Data Persistence Layer

AL systems involve dealing with fluctuating data. Therefore, a mechanism for storing the associated information is indispensable to ensure the traceability of labels. To address this need, a data persistence layer has been established in the LIFEDATA core framework, the central element of which is a relational database. To ensure the reliability and scalability of the database, LIFEDATA employs a container-based approach that isolates the database from project-specific dependencies.

**Object Relational Mapping.** Crucial in the data persistence layer is providing information on samples, labels, user information, and event logs, making it a key component in data provenance. A key element is the management of database interactions. This is achieved through the implementation of SQLAlchemy, an Object Relation Mapper (ORM) and Python SQL Toolkit (Bayer 2012). This abstraction toolkit facilitates database operations by leveraging the SQL expression language in conjunction with generative Python expressions.

**Listing 5.3:** Excerpts from the „Assignment Service “ implementation. The class „AssignmentService “ encapsulates the core functionality of sample assignment within the sample-annotator relationship. It is initialized with the „AssignmentRepository“, „SampleProjectRepository“ as well as „Event-Dispatcher“ and shows the integration of events within the method „get_sample()“.

```python
# lifedata/annotations/assignment.py
from .events import EventDispatcher
from .events import SampleAssigned
# ...

class AssignmentService:
    def __init__(
        self,
        event_dispatcher: EventDispatcher,
        sample_repository: SampleProjectRepository,
        assignment_repository: AssignmentRepository,
    ):
        self._events = event_dispatcher
        self._assignments = assignment_repository
        self._samples = sample_repository

    def get_sample(self, annotator: Annotator) -> Sample:
        # ...

        sample_params = self._assignments.query_for_sample_id(annotator)
        sample_id = sample_params["sample_id"]

        assignment =
        ↪  Assignment(sample_id=sample_id,annotator_id=annotator.id,)

        self._assignments.add(assignment)
        self._events.dispatch(SampleAssigned(sample_id=sample_id,
        ↪  annotator_id=annotator.id))
        return self._samples.by_id(sample_id)

# ...
```

Data modeling is likewise implemented as part of the data persistence layer. This involves defining the structure of the data within the database using SQLAlchemy's declarative base. Relevant data classes from the annotations domain layer map in the data persistence layer to a class of the database modeling, which in turn corresponds to a database table, in which each column corresponds to a class attribute. In Listing 5.4, we revisit the example of the assignment entity, where the code snippet now shows the data modeling of the data persistence layer.

The data model demonstrates how the data structure for the assignment is maintained down to the database level, with the attributes directly correlating to the database columns. This begins with naming the corresponding table in the database in line 6, and defining the columns in lines 8-11. While lines 9-11 correspond to the attributes of the `Assignment` data class in the Annotation Domain Layer, additional database specifications are made when modeling in the data persistence layer.

This code snippet furthermore shows how the relationships between the entities in the database are modeled. Line 13 establishes the relationship between database tables, in this example, the relationship between Assignment and the associated class `Sample`. This is a bidirectional relationship, meaning that in the `Sample` class, a corresponding property that refers back to the `Assignment` class and is named assignment must exist, enabling navigation and data access from both sides of the relationship. The resulting database schema of the relational database at the initialization of a LIFEDATA project is given in Figure 5.4, which provides an overview of the various tables along with their columns and relationships to each other.

However, the use of the ORM approach in LIFEDATA facilitates interac-

```
# lifedata/persistence/models.py
from .database import Base
# ...

class Assignment(Base):
    __tablename__ = "assignment"

    id = Column(Integer, primary_key=True, index=True)
    annotator_id = Column(String, index=True)
    sample_id = Column(String, ForeignKey("samples.id"), index=True)
    created = Column(DateTime, default=func.now())

    sample = relationship("Sample", back_populates="assignments")

# ...
```

**Listing 5.4:** Definition of the „Assignment" data model within LIFEDATA's Data Persistence Layer. The class „Assignment" specifies the attributes of the data class and inherits from „Base", the central database configuration. Each attribute corresponds to a column of the „assignment" database table.

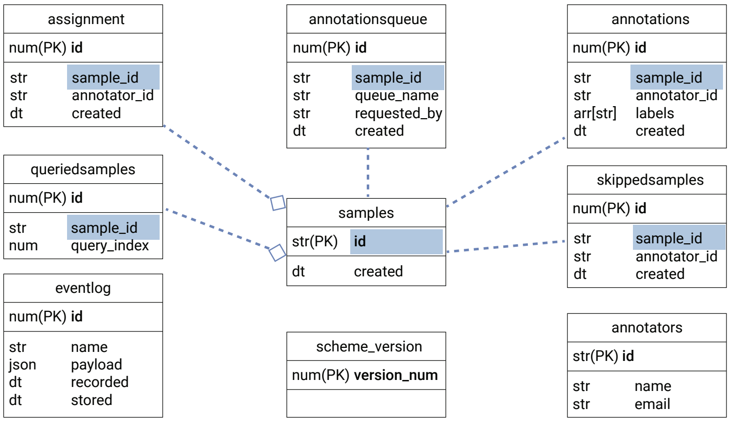


**Figure 5.4:** Initial database schema in LIFEDATA.

tion with the database and supports versioned database migration. The version control of the database schema is evident in the `scheme_version` table, allowing for schema modification during the project's duration to meet dynamic project requirements without interrupt the existing database structure and implementing changes in a traceable manner.

**Repository Pattern.** To achieve an abstraction of the data layer, the core framework adopts the repository pattern, decoupling data access in the data persistence layer from the rest of the logic. To this end, six different repositories are implemented, each tailored to a specific type of data or objects. Each of these repositories addresses a distinct entity: annotations, annotators, assignments, events, queryset, and samples.

Continuing with the example of assignments, we look at Listing 5.5. The implementation includes several methods, each relevant for specific operations related to assignments. For instance, there are methods for creating new assignments or retrieving specific assignments.

During instantiation, the `DBAssignmentRepository` repository is initialized with a database session. This session setup is detailed in a dedicated configuration, as discussed in the ORM section. Additionally, the initialization includes instances of `QuerySetDBRepository` and `MachineLearningService`. This configuration showcases the pattern of interactions among subsequent system components.

The method `_instantiate(...)` converts the implemented database model of the Assignment class into objects of the Annotation Domain

**Listing 5.5:** Excerpt of the extended „Assignment Repository“ implementation as part of the data persistence layer. The defined class „DBAssignmentRepository“ is initialized with references to the database session, „QuerySetDBRepository“ , and „MachineLearningService“. It includes methods for instantiating assignment objects of the corresponding dataclasses and for querying samples based on the annotator, integrating updates from the ML service as required.

```
 1 # lifedata/persistence/assignment_repository.py
 2 # ...
 3 class DBAssignmentRepository(AssignmentRepository):
 4 
 5     def __init__(
 6         self,
 7         db: Session,
 8         queryset_db_repository: QuerySetDBRepository,
 9         machine_lerning_service: MachineLearningService,
10     ):
11         self._db = db
12         self._queryset_db_repository = queryset_db_repository
13         self._machine_lerning_service = machine_lerning_service
14 
15     def _instantiate(self, model: models.Assignment) -> Assignment:
16         return Assignment(
17             annotator_id=model.annotator_id,
18             sample_id=model.sample_id,
19             created=model.created,
20         )
21 
22      def query_for_sample(self, annotator: Annotator) -> dict:
23 
24         sample_id =
           ↪   self._queryset_db_repository.query_for_sample(annotator)
25         # ...
26         if sample_id is None:
27             status = self._machine_lerning_service.ml_update()
28       # ...
29 # ...
```

Layer, concretizing the separation of these two implementations. The minimal excerpt of implementing the method `query_for_sample(...)` illustrates a crucial aspect of the sample-annotator assignment. If the existing queryset is empty, the ML service is called at this point to request new samples for annotation. The related logic for occurs in the project instance, which is connected to the core framework through the next layer.

### 5.3.4 Project Persistence Layer

To integrate LIFEDATA into any AL project, the Project Persistence Layer serves as a bridge between the core LIFEDATA framework and the external AL project, aiming to facilitate the integration of functionalities and interoperability between the AL project and LIFEDATA. This is achieved through a component containing functions for interacting with the acrshortML pipeline and handling artifacts.

At its core, the Project Persistence Layer includes functions for triggering or re-triggering the entire ML pipeline or parts of it. These functions are necessary in scenarios where new samples are required for labeling or when the ML service needs to be updated with the latest data annotations. An example of this is illustrated in Listing 5.5, where a relevant function is invoked in line 27 when a sample is requested for assignment to an annotator.

To synchronize the core framework with the acrshortML project, the Project Persistence Layer further defines functions for managing the artifacts associated with executing the ML service. This includes tasks such as exporting the current labelstate from the database and importing query sets, which include samples that have been selected by the QS and are ready for annotation.

The interface between the core framework and the project instance in this layer encapsulates these functionalities and simplifies the integration process into the AL project. While the logic for APIendpoints and data views is defined in the core framework, the implementation can be tailored to the specific AL project and is manifested, as shown in Figure 5.2, in the LIFEDATA Aonfiguration within the project instance.

## 5.4 Project Instance

In addition to the core framework, the second essential part in LIFEDATA is the project-specific structure, summarized as the LIFEDATA Project Instance in Figure 5.2. As depicted, the framework mandates that all code modules are managed through a source code version control system. The computed artifacts, such as the trained model or the generated query set for a specific data state, are managed by a data version control system.

A project instance represents the specific implementation of an AL project based on LIFEDATA. For their initial creation, LIFEDATA provides a project template, guiding developers in adapting the structure to their specific use case. The entry point for this process is demonstrated in Listing 5.1, which showcases the related CLI logic involving downloading the provided template and creating a new project.

LIFEDATA's project template employs Git (Chacon *et al.* 2014) as a distributed version control system, and DVC (Iterative 2020) is provided to synchronize the source code and resulting data versions. Using these two version control tools allows for adopting the development approach introduced in Chapter 4, thereby providing a foundation for tracking changes, managing branches, and facilitating collaboration among several engineers involved in the project.

Starting with the yellow-highlighted section of Figure 5.2, which reveals a project structure incorporating multiple components, the template provides functionalities for data import, the ML pipeline, and the annotation widget. Further components streamline the development, deployment, and management of a LIFEDATA project, and we will detail them in the following section.

### 5.4.1 Basic MLOps Components

Within the part managed by the Source Code Version Control System, there are code modules that serve primarily for the development and operation of the LIFEDATA project. In the reference architecture, they are arranged in a sequence, positioned directly below the core framework and above the data-processing components, with the LIFEDATA Configuration spanning three levels, illustrating the interrelation between the core framework and the AL project.

Section 5.3.4 indicates that project-specific implementations and functions are defined within this LIFEDATA Configuration, bridging the core framework and the AL project. The project template includes functions for retraining the model on a specific data version, which may occur, for instance, after new labels have been added to the training set following an annotation iteration. Moreover, the data import and export to and from the core framework are implemented on the side of the AL project, meaning that annotation information extracted from the core framework's database is imported into the AL project, while a function for exporting the created query set ensures that it is available for the components of the core framework upon creation. In other words, the LIFEDATA Configuration implements the specific logic that, for example, saves the labeling status exported from the annotation database into a file readable by the ML pipeline or triggers the corresponding execution of the ML pipeline when called.

The execution of individual elements of the ML pipeline is managed by the Orchestration Component, which coordinates the various modules. It thus represents a central part of the operational process within the project instance, ensuring that the sequence of operations is structured and timely. It schedules tasks and monitors their progress, while the CI/CD/CT Component is responsible for automating the creation of training, tests, and deploying the resulting artifacts. This includes both the software and AL-related artifacts, such as the query set and the trained ML model. Within the CI/CD/CT Component, instructions for the runners introduced in Section 4.2.3 can be defined, such as a nightly retraining of the ML model or performing its automated deployment.

Another component of the project template includes the Infrastructure Utilities, which encompass tools and scripts necessary for the infrastructure-related tasks of a project. This consists of the provisioning, monitoring, and scaling resources, such as GPU usage. They are always highly dependent on the hardware or cloud platform used and, therefore, require customization for the specific AL project, just like the libraries for model training.

Furthermore, the LIFEDATA project template includes a Package Management realized through virtual environments in Anaconda for the project's software dependency management (Anaconda Inc. 2016). This ensures the system is built with the correct versions of libraries and packages, guaranteeing its consistency and reproducibility across different environments and platforms.

### 5.4.2 Data Import

Adjacent to the code module area highlighted in yellow in Figure 5.2, there lies the section managed by the data version control system, colored in blue. This system tracks various data stages, from raw data through computed artifacts to extracted label sets, which are interim stati of data annotations.

As mentioned in Section 4.2.1, the integrated data version control system DVC operates on the principle of computing hashes for managed files and directories and storing these as version numbers in configuration files. For this purpose, raw data are held in a storage system and represented within the project by placeholder files, which, in addition to the computed hash values, further define specifications, dependencies, and output entries. Since these placeholder files are managed by Git, versioning of both code and data states is always ensured, even when the actual datasets are not checked into the project.(Iterative 2020)

This feature allows, through the integration of DVC within a LIFEDATA project, to interlink multiple repositories. DVC refers to this concept as a „Data Registry“, a kind of data management middleware, by setting up a repository for versioning datasets that contain the required metadata and their complete change history (Iterative 2023a). The LIFEDATA project template adopt this concept, outsourcing tasks that typically occur in the data development pipeline with ETL processes to a separate repository, which is tracked by Git and DVC. This approach offers several advantages, such as making these data available for other projects, reducing the complexity and dependencies between individual projects, and, in line with Section 3.3.1, utilizing cleaned data in the AL project.

For importing data into the LIFEDATA project, DVC provides various options, enabling data to be loaded from different storage systems using protocols such as S3 or SSH. These are made available in adapters, thereby supporting a range of cloud providers and self-hosted storage types (Iterative 2023b). DVC downloads the target files or directories and tracks them within the LIFEDATA project. If the data source changes, for instance, new unlabeled data is added to the project, these can be updated at a later stage - a process that might occur either manually or during each iteration of the ML pipeline's execution.

Once the data is checked in, it becomes available in the LIFEDATA project as the initial Raw Data stage and can be consumed by the ML pipeline.

### 5.4.3 Machine Learning Service

An ML pipeline typically consists of individual steps that are executed in a specific order. It is a proven practice to divide these steps within a pipeline (O'Leary *et al.* 2020), ensuring the structured execution of individual tasks while considering their dependencies. The ML pipeline will be formed by making each stage dependent on another, i.e., defining the output of one stage as the input for another.

This principle is followed in the LIFEDATA project template. The ML pipeline defined therein begins with separate steps for data preparation, model training, querying, as well as evaluation and validation, as can be seen in Figure 5.2. Each step of the ML pipeline is initially abstractly implemented, thereby being independent of the underlying ML problem and -framework.

Reflecting the code principles described in Section 4.2.2, the various steps of an ML pipeline are commonly realized through individual code modules configurable via parameters. Typically, each stage takes in data, executes the code using the provided parameters, and produces an output as artifacts.

By integrating DVC (Iterative 2020), the ML pipeline in a LIFEDATA project can be internally represented as a graph, with nodes being stages and edges being directed dependencies. Representing the ML pipeline as a Directed Acyclic Graph (DAG) allows for implementing systems with a sequential execution flow. Eliminating repetitions between ML pipeline

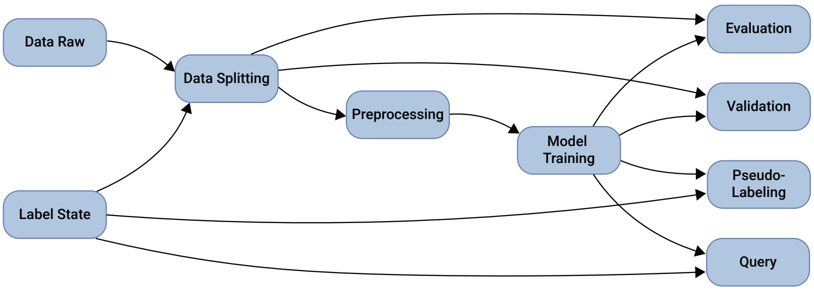


**Figure 5.5:** Initial ML pipeline definition provided as a Machine Learning Service in an LIFEDATA project instance. The nodes of the DAG represent the stages of the ML pipeline, and the edges indicate their dependencies.

stages avoids loops between two steps, implying the topological property of the DAG.

This architecture allows for the modularization of the ML pipeline on the one hand. Due to its linear ordering, it requires that there can be no circular relationships between one or more ML pipeline steps on the other. For this purpose, the ML pipeline is provided in a LIFEDATA project as a ML service that triggers the execution of the required modules by the orchestration component. The definition of the original ML pipeline, including all its stages and dependencies, is illustrated in Figure 5.5. The nodes of the DAG represent the stages of this ML pipeline, while the edges display their dependencies on each other.

Starting with the stages of „Data Raw“ and „Label Set“, which each checks the version of the database and annotation statuses and generates a report, follows the phase of data splitting. In addition to the otherwise typical proportional division into training-, test-, and validation-data, the LIFEDATA project template provides methods for static splitting to ensure that the test dataset is not used for training the model throughout the entire project phase. This is achieved by generating and versioning indices for the respective datasets.

The arrangement of data preprocessing after data splitting enables the separate preprocessing of training-, test-, and validation-data. For example, the vectors required for normalization are calculated in isolation on the respective datasets, thus avoiding data leakage and enabling realistic simulations through statistical independence, even if the datasets change throughout the AL project.

Subsequently, the model training stage is defined, which depends on the preprocessed data and transitively on the phase of data division. The stages are directly connected as an additional dependency to enable the evaluation and validation of a trained model on the test data. Creating non-transitive relations between the ML pipeline stages of label states to query and pseudo-labeling enables their execution independent of retraining the model. Thus, a model version can be in use for an arbitrarily long time.

To gain a more detailed understanding of the implementation of the ML pipeline in a LIFEDATA project, the query module serves as a predestined running example.

Listing 5.6 demonstrates the relevant part of the ML configuration. Here, a separate class is defined for each stage of the ML pipeline. It specifies the available parameters and defines paths for code and data artifacts. If certain parameters should be associated with a submodule, it is advisable to follow this structure and define them in an inner class. Parameters that apply across multiple stages, on the other hand, are set outside of each class, as demonstrated by the example of `RANDOM_STATE`. When used correctly, this example ensures that imported libraries, such as NumPy (Harris *et al.* 2020), scikit-learn (Pedregosa *et al.* 2011; Buitinck *et al.* 2013), TensorFlow (TF)(Martín Abadi *et al.* 2015), and PyTorch (Paszke *et al.* 2019), are initialized with the same random state, even when they are called from different modules in LIFEDATA.

**Listing 5.6:** Example configuration of the query stage.

```python
# mlconfig.py
from pathlib import Path

root = Path(__file__).parent.absolute()

RANDOM_STATE = 123

def absolute_path(relative_path: str) -> Path:
    return root / relative_path

#...

class QUERY:
   QUERY_SET_FILE = "data/query/queryset.csv"
   QUERY_BATCH_SIZE = 200
   QS_STRATEGIES = ["uncertainty", "random"]
   QS_RATIO = [2,1]

   class UNCERTAINTY:
      CONFIDENCE_THRESHOLD = 0.9
      #...

#...
```

This implementation allows for access to a unified configuration from each code module along the ML pipeline. For example, in the code module of the Query stage, the value for the size of the queryset, namely the number of samples selected for annotation by the QS, can be accessed through `mlconfig.QUERY.QUERY_BATCH_SIZE`, as set in the configuration file. The source code's version control system enables systematic comparison of different configurations and settings, forming the basis for consistent reproducibility of the resulting artifacts.

Another aspect regarding computed artifacts, evident from Listing 5.6, is the handling of paths within the LIFEDATA project instance. The project structure proposed in 4.2.2 suggests creating a dedicated location for data artifacts within the project, parallel to the code modules for each stage of an ML pipeline. In the ML configuration, the paths are defined along with an appropriate method for determining the absolute path in relation to the configuration file. This approach allows for platform-independent execution without the need to adjust the path specifications in the code modules, for instance, both on the developer's client and in the CI/CD/CT runner that, for example, executes the ML pipeline in a cloud instance.

As mentioned, it is necessary to define all tasks in all stages and their dependencies and configurations within the ML pipeline to enable their automated execution. This definition is managed by DVC (Iterative 2020), specifying inputs and outputs at each step as well as potential dependencies, thus ensuring that the creation of all artifacts is transparent and reproducible. Some use cases may require adjustments to the original

ML pipeline design from Figure 5.5, such as adding or removing certain stages. To achieve this flexibility, these stages and their dependencies can be identified and adjusted to the definition of the ML pipeline by the data scientist.

We continue with the example of the Query module as an essential part of the ML Service. It facilitates the selection of samples for annotation and enables the automatic generation of query sets. It uses one or more QSs to select samples whose annotation is intended to lead to the desired model change. Typically, QSs utilizes the ML model. Hence, their execution occurs post-training. This condition is illustrated in Listing 5.7, which shows the relevant excerpt from the ML pipeline definition.

**Listing 5.7:** Query stage in ML pipeline definition.

```yaml
vars:
- mlconfig.py

stages:
   #...
   query:
      cmd: python myproject/query/query.py
      deps:
      - ${LABEL_STATE.UNLABELED_FILE}
      - ${LABEL_STATE.LABELED_FILE}
      - ${MODEL_TRAINING.MODEL_FILE}
      - myproject/query/
      outs:
      - ${QUERY.QUERY_SET_FILE}
      params:
      - mlconfig.py:
         - QUERY

   #...
```

The described structure is as follows: Given a valid ML configuration, the individual stages are defined. The command to be executed is specified, as well as the associated dependencies in the form of paths to code and data artifacts. In the case of the Query module, this involves files containing information about the samples, regardless of whether they are already labeled or unlabeled. Moreover, the trained model and the entire project folder of the corresponding Query module, containing all code files, are required.

The calculated artifacts consist of a list of samples the QS selected and stored under the corresponding output path. Additionally, the class of the relevant phase from the ML configuration is specified. If any of the specified artifacts produced by a particular stage change, or if a reconfiguration requires the recreation of individual artifacts, the relevant parts are calculated and stored by triggering the Machine Learning Service. Thus, changes can be, for example, a new model version or a change to the source code or its configuration. As described, this effect is achieved through the DAG architecture in combination with the integrated data versioning system. It encompasses all modules across the entire ML pipeline in the LIFEDATA project instance.

In addition to the query stage, which generates the set of samples to be labeled, the modules for generating pseudo-labels, as well as for evaluation and validation, are defined as possible end steps of the ML pipeline. While the step of generating pseudo-labels enables the realization of semi-supervised learning scenarios, the step of evaluation is intended for calculating model metrics. For a deeper analysis of the trained model, the implementation of XAI methods is placed in the validation phase. These methods are meant to facilitate the interpretation of model predictions.

The architecture of the ML pipeline in the LIFEDATA project instance ensures that the artifacts created by these steps, whether they are calculated model metrics or generated model interpretations, always depend on the versions of the respective models and training data. Consequently, these outputs, further tracked by the data version control system, can be used for a UI or reporting purposes.

### 5.4.4 Webinterface Components

As described in Section 5.3, LIFEDATA provides web-based GUIs for domain experts. Their project-specific implementation is heavily dependent on the use case and summarized in the reference architecture of the AL project in Figure 5.2 as „Webinterface Components“ .

**Annotation Widget.** The annotation of samples by a human oracle within a web-based GUI allows for easy and efficient labeling of samples by multiple annotators. To ensure that LIFEDATA is agnostic to labels and data types, an Annotation Widget has been designed to enable Data Scientists to implement a wide range of pool-based AL scenarios.

The connection between the Annotation Widget in the project instance and the core framework is realized through the REST-API, the methods of which are listed in Table 5.3. The provided Backend Service offers an interface for receiving labels and transmitting samples to be displayed in the Annotation GUI. Thus, seamless communication between the project instance and the core framework is ensured. The flexibility in customizing the rendering process ensures that users can tailor their annotation process to their specific requirements.

To achieve runtime-performant yet highly flexible web-based frontends, as needed for annotating training data, LIFEDATA offers an out-of-the-box implementation based on the JavaScript library React (Meta Open Source 2013). The rendering for the respective data type, as well as labeling tools such as class selection or provision of tools for measurement or marking, can be implemented within the Annotation Widget in the project instance itself or replaced by a third-party app.

**Dashboard.** As an additional web-based GUI, the AL project in LIFEDATA offers a dashboard. This component is designed to consolidate all relevant information and provide domain experts with key metrics about the learning progress of the ML model, as well as crucial indicators of the current status of the annotation process.

The modular architecture of the ML pipeline, with its evaluation and validation phases, is essential for the implementation of monitoring and reporting. For the straightforward implementation of the dashboard component in LIFEDATA, the integration of the experiment tracking service MLflow is foreseen. MLflow simplifies the process of tracking experiments, parameters, and results, thereby enhancing the system's overall monitoring capabilities. This is accomplished through a metric tracking API provided for Python. The metrics calculated in the ML pipeline are sent to the MLflow server, which consists of a relational database and a web interface.(Zaharia *et al.* 2018)

For more complex use cases, the artifacts tracked by the data versioning tool can themselves be utilized to display in a custom UI. With the integration of DVC (Iterative 2020) into LIFEDATA, a Python API is further available for reading results stored and tracked during the execution of the ML pipeline, such as a table-file with performance metrics of the ML model from a defined remote storage. These can then, for example, be used for a custom UI implementation.

## 5.5 Limitations

While LIFEDATA provides a structured approach to managing AL projects, it's essential to acknowledge its inherent limitations to provide a balanced and realistic understanding of its applicability and areas for potential enhancement.

A primary limitation of LIFEDATA's design is its suitability primarily for pool-based AL scenarios. It excels in environments where a large pool of unlabeled data is available for selective querying but is not inherently designed for stream-based scenarios where data arrives sequentially, and decisions about labeling must be made on-the-fly. Adapting LIFEDATA for stream-based scenarios would require significant modifications to its underlying architecture and workflow, including real-time decision-making capabilities and a more dynamic model updating process.

Through the integration of DVC (Iterative 2020) in the project template, LIFEDATA employs a DAG architecture for its ML pipeline, which presents its own challenges while offering several advantages in terms of modularity and clarity. The DAG architecture facilitates a clear and manageable sequence of operations, ensuring that each step is logically ordered and dependencies are maintained. This structure is particularly beneficial for simpler ML pipelines where the flow from data preparation to model training and evaluation is relatively linear and straightforward. However, as ML pipelines become more complex and involve numerous interdependent steps, the DAG architecture can become a limitation. Complex pipelines with multiple stages or concurrent processes can be challenging to represent and manage within a strictly linear DAG structure. Furthermore, this limitation impacts the parallelization of tasks. While DAGs can inherently handle some parallel operations, the rigid, non-cyclic nature of DAGs means that they might not efficiently utilize all available computational resources or optimally parallelize tasks that have complex interdependencies, posing a scalability concern. For more complex projects, alternative techniques for data version control, supplemented by additional tools as orchestration components, may be required.

Furthermore, the engineers using LIFEDATA's web-based GUIs must handle the deployment. This involves setting up a server- or cloud environment and ensuring fulfillment of CI/CD principles, security, and compatibility across different platforms and devices. The deployment process can be complex and time-consuming, requiring users to develop web deployment expertise.

Setting up a LIFEDATA project, despite the provided project template, therefore involves considerable effort. The template offers a kick-start by establishing a structured environment with predefined components and workflows. However, the complete integration and customization to a specific ML problem, as well as the implementation of an annotation interface, require substantial upfront work. While this flexibility allows the framework to be tailored to specific tasks and requirements, meaning that a significant level of implementation is necessary at the beginning of an AL project with LIFEDATA.

This generalization vs. specialization dilemma is a crucial aspect: While the project template supports prominent ML frameworks like TF (Martín Abadi *et al.* 2015), PyTorch (Paszke *et al.* 2019) and frameworks who follow the Scikit-Learn-API(Pedregosa *et al.* 2011), the model code must be implemented from scratch, and the same applies to the data preprocessing and QSs included in the template. Their logic needs to be implemented, whereas other AL frameworks provide a range of common QSs out-of-the-box. While this makes LIFEDATA highly agnostic and usable for different data formats, it requires a significant amount of implementation by the data scientist using it.

Moreover, the LIFEDATA framework's concepts are primarily tailored to classification problems and do not inherently support sequence labeling or regression tasks. This specialization means that while it supports a range of classification scenarios, such as binary or multi-label classification, its applicability to other types of ML tasks is limited. Users looking to tackle sequence labeling, regression, or other non-classification tasks would need to significantly customize or extend the framework, potentially undermining the benefits of its out-of-the-box functionality.

Despite these limitations, LIFEDATA has a critical advantage: by integrating concepts of traceability, reproducibility, and explainability, it paves the way towards trustworthy AI development with an AL loop. The following Chapter 6 presents two real use cases in which the framework concept is applied, demonstrating the feasibility of AL projects with LIFEDATA.

## 5.6 Summary

The LIFEDATA framework presented in this chapter has been developed to implement AL projects in the spirit of AI development by incorporating transparency and traceability. LIFEDATA includes technical components for code and data versioning that enable users to comprehensively trace the learning process from selecting training data to interpreting model results in an AL project.

The framework further includes multiple interfaces, a relational database for storing annotations including their origins, and a web-based user interface that facilitates data annotation by human oracles.

This framework consists of open-source software implemented in Python and available on GitHub. It is divided into two main parts: a core framework and a project template, which serves as a starting point for new projects and as a template for implementing AL projects. Both components are designed to be data- and algorithm-agnostic, supporting a wide range of use cases.

# 6 Use Cases & Evaluation

**6.1 Skin Image Analysis . . 148**
6.1.1 Data Basis . . . . . . . . . 148
6.1.2 Application . . . . . . . . 151
6.1.3 Active Learning Simulation . . . . . . . . . . . . . 157
**6.2 ECG Signal Classification . . . . . . . . . . . . . 161**
6.2.1 Data Basis . . . . . . . . . 161
6.2.2 Application . . . . . . . . 164
6.2.3 Active Learning Simulation . . . . . . . . . . . . . 174
6.2.4 Human-in-the-loop Study 179
**6.3 Summary . . . . . . . . . 182**

To establish the connection between the LIFEDATA framework introduced in Chapter 5 and the development methodology discussed in Chapter 4, this chapter investigates the technical feasibility of applying the framework in two specific use cases within the life sciences. The life sciences provide an ideal testing ground, as they are typically associated with high risks and underline the need for trustworthy AI. Moreover, this domain highlights specific challenges that are characteristic of AL projects:

a) **Imbalanced Data.** There is an unequal distribution of samples across different classes. Related challenges potentially affect the model and the annotation process, as the majority classes dominate the dataset. (Ertekin *et al.* 2007b; Ertekin *et al.* 2007a; Aggarwal *et al.* 2020)
   **Section 6.1.3** addresses the challenge of unbalanced datasets by using a dataset for simulations that is dominated by a majority class. This illustrates the impact on model training and annotations in an imbalanced class context.
b) **Costly Annotations.** A high level of qualification is required to annotate the data, often involving a long period of education for the annotator. This circumstance additionally increases the labeling costs, since the time of the human oracle represents a rare resource. (Freeman *et al.* 2021; Bernhardt *et al.* 2022)
   **Section 6.2.3** addresses this issue by extending the query module to examine annotation efficiency in a multi-label classification context, which further increases the complexity and, thus the effort required for annotations.
c) **High Quality Labels.** In addition to the specialized knowledge of domain experts, the degree of accuracy regarding the assigned annotations significantly influences the model's performance in the real-world. In many cases, the quality of the resulting model is directly tied to the quality of the labels it is trained on. (Alonso 2015; Northcutt *et al.* 2021)
   In **Section 6.2.4** this is emphasized by a study with human annotators that examines self-agreement and agreement between annotators as a indicator of annotation reliability.

Furthermore, this chapter explores the key aspects of the technical implementation and adaptability of the LIFEDATA framework in the aforementioned projects, emphasizing its significance for ensuring the integrity of ML artifacts. This examination serves as a demonstration of how the framework, as discussed in Chapter 5, as well as concepts presented in Chapter 4 can be applied effectively in real-world scenarios.

These two use cases were carried out within the LIFEDATA project, funded by the BMBF[1]. Parts of the results were published in Stieler *et al.* 2023b.

[1] Reference number 031L9196B

## 6.1 Skin Image Analysis

Skin cancer, particularly melanoma, continues to be a type of cancer with a globally high incidence rate, which is on an increasing trend (Riker *et al.* 2010). Early detection significantly improves the chances of survival, and as a result, the use of AI-based skin image analysis has gained considerable attention in the research community in recent years. For instance, the International Skin Imaging Collaboration (ISIC) has conducted various ML Challenges, during which promising results for AI-based skin image analysis were published (Celebi *et al.* 2023).[2]

The analysis of images of skin lesions, anomalies of the skin that manifest in various forms of changes, can involve different steps, each subject to extensive research. These include preprocessing and feature extraction as well as segmentation to determine relevant image areas, for example, delineating the affected lesion on the skin.(Li *et al.* 2022)

The third crucial step for diagnostic support is the classification of skin lesions (Brinker *et al.* 2018; Adegun *et al.* 2021). While traditional classification methods use pixel- or region-based techniques to extract features, which are then processed by classifiers like Support Vector Machines, recent developments in AI research demonstrate that the use of DL models, particularly CNNs (Haggenmüller *et al.* 2021; Shetty *et al.* 2022), in processing and classifying images of skin lesions has proven to be superior to traditional methods, and even to dermatologists in specific classification tasks (Esteva *et al.* 2017; Tschandl *et al.* 2020).

Despite the progress, there are still several challenges (Goyal *et al.* 2020). One of these is the limited amount of available, high-quality, and labeled data for training such DL models for skin lesion classification. Thus, the following case study on skin image analysis focuses on simulating the AL workflow.

The goal is to investigate whether the annotation effort in a scenario of a DL-based skin lesion classifier can be reduced and thus make data labeling more efficient. For this purpose, we simulate the human oracle by providing the true label through a lookup in an already labeled dataset, which is explored in the following section.

### 6.1.1 Data Basis

Given the considerable attention paid to applications in the field of AI-based skin image analysis, numerous initiatives have emerged in the past that have compiled various publicly accessible datasets. The systematic review by Wen *et al.* 2022 presents a comprehensive evaluation of these resources and emphasizes their significance as benchmarks for algorithm performance comparisons.

One of the datasets introduced is HAM10000, a collection of dermatoscopic images created as part of the „Human Against Machine“ research (Tschandl *et al.* 2020) to overcome the limitations of earlier collections often characterized by small size and limited diversity, focusing on specific types of lesions.(Tschandl *et al.* 2018)

While some datasets concentrate on specific lesion types, HAM10000 encompasses 10,015 images of various lesions categorized into seven

[2] An overview is available by the published results from the ISIC Challenges: https://challenge.isic-archive.com

| Class | Description |
|---|---|
| NV | *Melanocytic nevi*, benign neoplasms of melanocytes, is by far the most prevalent group in the dataset. |
| MEL | *Melanoma*, a malignancy from melanocytes with various forms. |
| BKL | *Benign keratosis* groups similar biological conditions like seborrheic keratoses and solar lentigo, despite diverse dermatoscopic appearances and some resembling melanoma. |
| BCC | *Basal cell carcinoma*, a prevalent form of skin cancer. |
| AKIEC | *Actinic Keratoses* are treatable, noninvasive squamous cell carcinoma variants, often precursors to more serious forms of skin cancer. |
| VASC | *Vascular skin lesions* in the dataset include a range of diseases. |
| DF | *Dermatofibroma* is a benign skin lesion typically presenting with peripheral reticular lines and a central white patch indicating fibrosis. |

**Table 6.1:** Class description of lesions within the HAM10000 dataset (Tschandl *et al.* 2018).

| Variable | Description |
|---|---|
| lesion_id | Unique identifier for each lesion |
| image_id | Unique identifier for each image |
| dx | The diagnosis, described in Table 6.1 |
| age | The age of the patient |
| sex | Information on the gender of the patient |
| localization | Cathegorical indication of the lesions' location on the body |

**Table 6.2:** Provided metadata within the HAM10000 dataset (Tschandl *et al.* 2018).

groups. These categories are presented in Table 6.1. The images were collected over a period of 20 years at locations in Austria and Australia. The acquisition process involved extracting images and metadata from various sources and originating from different medical diagnostic devices. To determine the ground truth, dermatologists applied various diagnostic techniques.(Tschandl *et al.* 2018)

Before the dataset's publication, the images underwent manual histogram correction to optimize quality (Tschandl *et al.* 2018). Figure 6.1 illustrates three randomly selected samples. All images are in JPG format with a resolution of 450x600 pixels in the Red-Green-Blue(RGB) color space. Furthermore, the dataset contains metadata, the characteristics of which are outlined in Table 6.2. It indicates that multiple images may exist for a unique lesion.

Figure 6.2 displays the frequency of observed classes, the distribution between lesions, and the number of images for a unique lesion in the HAM10000 dataset. A notable overrepresentation of the class *nv* is evident, allowing for the investigation of QSs for frequent and rare classes concerning simulated data annotation.

The metadata exploration is further visualized as statistics in Figure 6.3, containing information on the population's demographic. Age data is

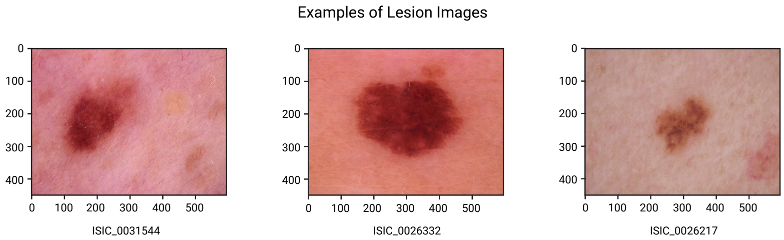


**Figure 6.1:** Examples of lesion images from the HAM10000 dataset published by Tschandl *et al.* 2018. The coordinates indicate the dimensions of the original sample shape of 450x600 pixels.

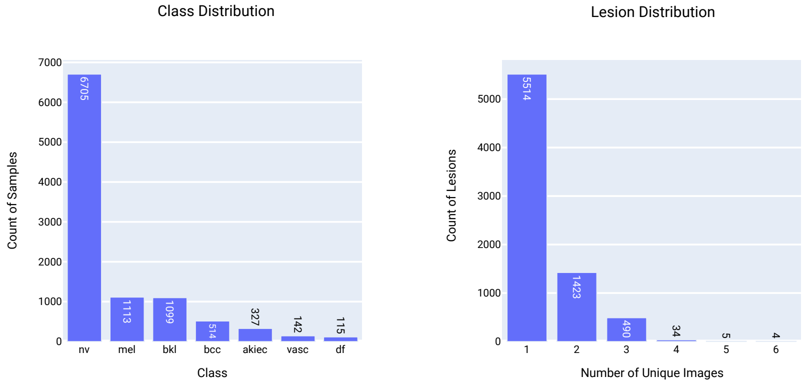


**Figure 6.2:** Class- and lesion distribution of the samples within the HAM10000 dataset (Tschandl *et al.* 2018). As evident from the count of samples, there is a significant imbalance, with the Nevus class being overrepresented.

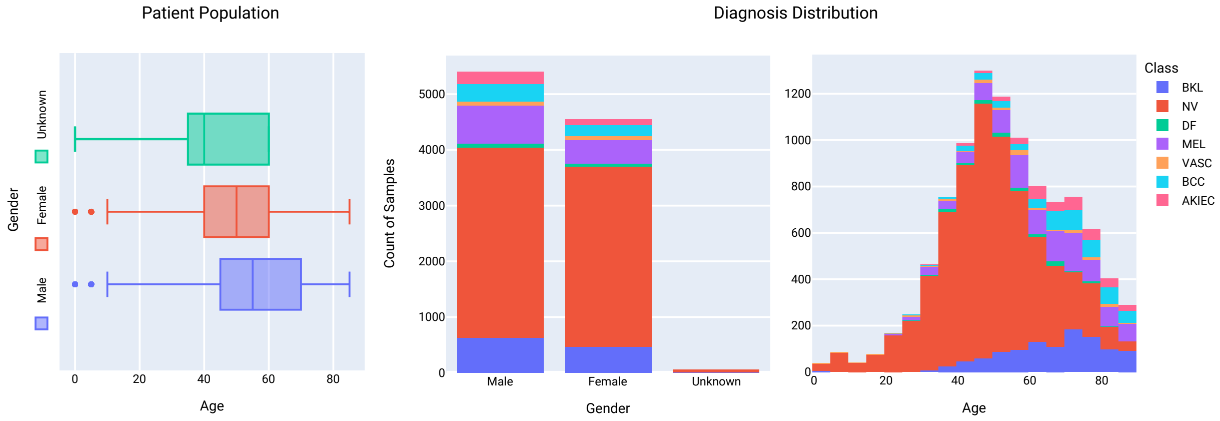


**Figure 6.3:** Distribution of samples in relation to the characteristics of the patient population of the HAM10000 dataset (Tschandl *et al.* 2018). While there appears to be no imbalance in the distribution of classes, there is a noticeable slight underrepresentation of female patients and a distribution of data across the entire lifespan.

categorized in 5-year intervals, covering the entire lifespan. The median age is 55 years for male individuals, 50 years for females, and 40 for persons without gender disclosure.

Although the overall number of women is slightly underrepresented, the distribution of diagnoses regarding gender and age seems balanced and shows no significant correlation. However, the authors note that the dataset includes data from various population groups, with no further details on ethnic groups and skin types. These limitations are relevant to the results' generalizability, as discussed by Wen *et al.* 2022.

Nevertheless, the dataset is suitable for investigating an AL project with LIFEDATA: It has proven to be a comprehensively examined dataset, already used in several ISIC Challenges (Cassidy *et al.* 2022) as training data, ensuring comparability of ML pipeline performance. Moreover, the class imbalance presents a realistic scenario for investigating AL and thus provides a solid basis for simulating an annotation process.

## 6.1.2 Application

The following chapter describes the application of LIFEDATA to the use case of skin image analysis, utilizing the previously described data foundation. The detailed description of adjustments to the LIFEDATA project instance (refer to Section 5.4) is provided by illustrating the project setup and offering a detailed overview of the ML pipeline implementation.

### Project Setup

In the implementation of the skin image analysis use case, LIFEDATA's project template was utilized and supplemented with components according to its reference architecture, as shown in Figure 5.2, including the MLOps components described in Section 5.4.1 that were not originally included. Furthermore, an infrastructure was established that adheres to the principles outlined in Chapter 4.2.3. The architecture of this infrastructure is depicted in Figure 6.4, including a high-level representation of each component and their interaction.

As a core system, the SCM application, a GitLab instance, houses the code repositories for the ETL project and the LIFEDATA AL project per the project template. Within these repositories, all code modules and data placeholder files are version-controlled. Further included is the CI/CD/CT component, which primarily defines the instructions for the CI/CD/CT runner. Initially, a job is set up to create a Docker image, which, upon completion, is uploaded as a build artifact to a dedicated image repository. Subsequent stages of the CI/CD/CT component, defined as secondary, utilize this to execute jobs for running the ML pipeline in a containerized environment on the CI/CD/CT -Runner.

The CI/CD/CT runner is installed on a high-performance computer equipped with two Intel Xeon Silver 4110 (Skylake-EP) 14-core CPUs, 96 GB RAM, one TB SSD, and twelve TB HDD local storage, as well as two NVIDIA Quadro P6000 GPUs with 24 GB each. As previously discussed, the CI/CD/CT runner executes the ML pipeline within a

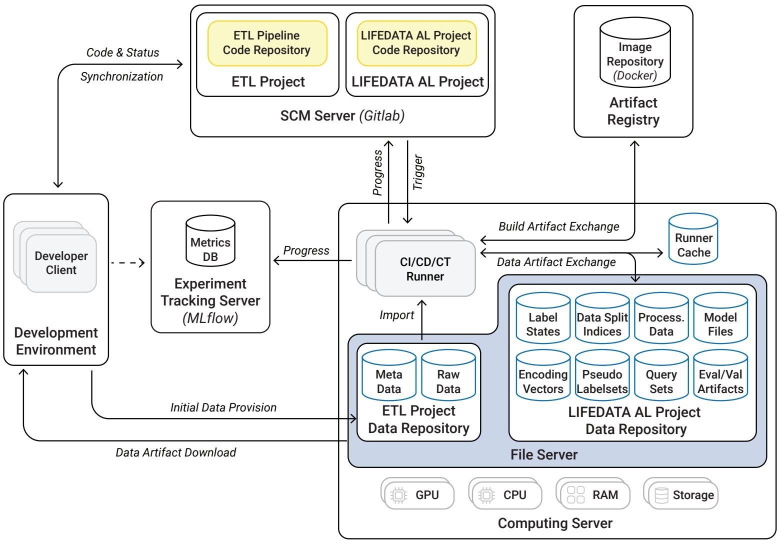


**Figure 6.4:** Project setup for the skin image use case. The figure illustrates a process and system perspective as a fusion between the components of the LIFEDATA reference architecture (Figure 5.2) and the applied infrastructure concept as described in Figure 4.3, showcasing the interaction of the individual elements within the specific LIFEDATA project instance.

Docker container, to which a specific directory is mounted as a local cache for the data artifacts.

DVC manages this cache as content-addressable storage to optimize data management and differentiates between caching individual files as well as directories, the latter of which can contain multiple files. Filenames and directory structures are transformed within these caches: files are renamed based on their content, and directories are flattened to a single level. This is done by generating a computed hash value uniquely representing the content.(Iterative 2024)

Further hosted on the computing server were the data repositories of both projects. These are specifically storage spaces of a file server used for distributed storage of various data artifacts. Similar to how the SCM server serves as a Git remote for code artifacts, these storage locations were configured as data remotes within the data versioning application. Following the concepts described in 4.2.3, the jobs within the CI/CD/CT component designated for executing the ML pipeline were defined such that all resulting data artifacts, such as the trained model and the query set, were pushed to the file server after execution and the updated data placeholder files were pushed to the corresponding branch of the SCM application as an commit.

Since the storage format within the file server structure is analogous to that of the DVC-managed runner caches, the data artifacts to be imported by other jobs from different CI/CD/CT runners if unavailable in the runner cache. For example, the individual iterations of the AL simulations were executed sequentially in separate CI/CD/CT jobs, whereby the status of the label state from the previous iteration could be accessed by loading the corresponding index file from the data remote. Additionally, developers involved in the project could access the data managed by DVC at a common location to download it into their local environment.

### Machine Learning Pipeline

The definition of an ML pipeline suggested in the project template was adapted for the skin image analysis use case, as illustrated in Figure 6.5, which is again illustrated the implemented ML pipeline as DAG, according to LIFEDATA's architecture of the Machine Learning Service. Since no semi-supervised scenario was implemented, the stage intended to create pseudolabels has been removed. Another significant change compared to the initial ML pipeline definition of LIFEDATA is the addition of a new stage, namely 'Encoding', which is placed between the model training and the query stage. The specific implementations in each stage are presented below to give a more detailed understanding of the design of the ML pipeline.

**Data Raw.** As described in Section 5.4.2, the project template from LIFEDATA suggests outsourcing the raw data used to a separate project where data engineering tasks occur. In simple terms, these tasks encompass the ETL process. Contrary to a real-world scenario, the data from the HAM10000 dataset do not require any particular transformation, as they are intended for research use and are already in a uniform format without any damaged samples.

To demonstrate the adaptability of the LIFEDATA project instance, the HAM10000 dataset, in its raw state, was nonetheless outsourced to a separate repository. Following this concept, in this implemented ML pipeline stage, the placeholder file for the version of the raw dataset is specified. This ensures that artifacts calculated during the project runtime by downstream stages of the ML pipeline remain consistent as long as the raw data or metadata do not change, and thus, the hash value specified in the placeholder file remains valid.

**Label State.** The annotation process was implemented through a code module in the experiment utilities, designed to simulate a human oracle in a real-world scenario. Upon successful execution of the query module within the Machine Learning Service, and consequently the entire ML pipeline, the labels for samples selected by the designated strategy were retrieved from the provided metadata. For recording the labeling status of each sample, the *Label State* stage, furnished by LIFEDATA, was utilized.

Within this stage, the index files, representing pools of annotated and unannotated samples, were modified by the code module's logic, operating independently from the ML pipeline managed by the data versioning system. This modification of the input data (new labels) necessitates re-computing subsequent stages in the ML pipeline, contingent upon activating the Machine Learning Service via a specific call. This mechanism represented individual iterations within the AL cycles.

**Data Splitting.** To ensure a reliable assessment of QSs and the model, the training dataset was proportionally sampled as follows: 70% of the samples were used for training the model. During the training process in the *Model Training* stage, 10% of the data were allocated for validation to prevent potential overfitting using early stopping methods. The remaining 20% of the data were reserved as a test dataset for evaluating the model's performance.

Since multiple samples from a single lesion are present in the HAM10000 dataset, the sampling method of the project template was extended for this use case. A specific logic ensured that different samples of the same lesion_id were grouped before being divided into training-, validation-, and test-datasets. This ensures that samples from the same lesion are kept together and prevents test set leakage, where a sample from a lesion is assigned to different datasets. As a result of this ML pipeline stage,

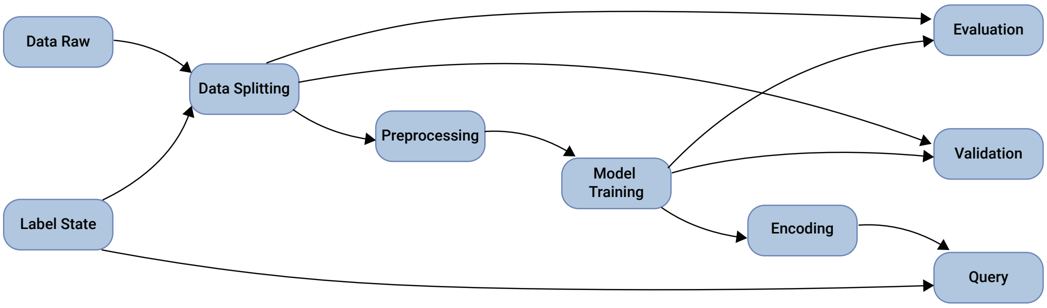


**Figure 6.5:** ML pipeline definition of the LIFEDATA project instance applied to the skin analysis use case. The initial structure has been adapted by removing the stage for semi-supervised learning. A separate stage for encoding has been added.

index files for the three datasets were created, ensuring consistent data sampling across multiple AL iterations.

**Preprocessing.** In the preprocessing stage, logic was implemented to scale the provided image files to a desired size. The chosen model architecture, described in more detail in the next stage, required an input size of 224x224 pixels, which is why these parameters were applied in the conducted experiments. The resulting artifacts of the code module within the preprocessing stage, concretely resized images using the 'crop' method, were stored. This meant that this computation step could be skipped in different experiments and across multiple AL iterations with the same configuration by retaining the already preprocessed image files in the subsequent stages of the ML pipeline. This approach significantly improved efficiency by eliminating the need to repeat the preprocessing step.

**Model Training.** In this stage, the preprocessed data were loaded into the model. The DenseNet101 model architecture (Huang *et al.* 2017) was implemented, backed by its application in numerous other studies (Jeong *et al.* 2023). The settings for batch size, number of epochs, and learning rate were parameterized, and optimization in this context yielded an optimal configuration of `BATCH_SIZE=8` and `LEARNING_RATE=0.001`.

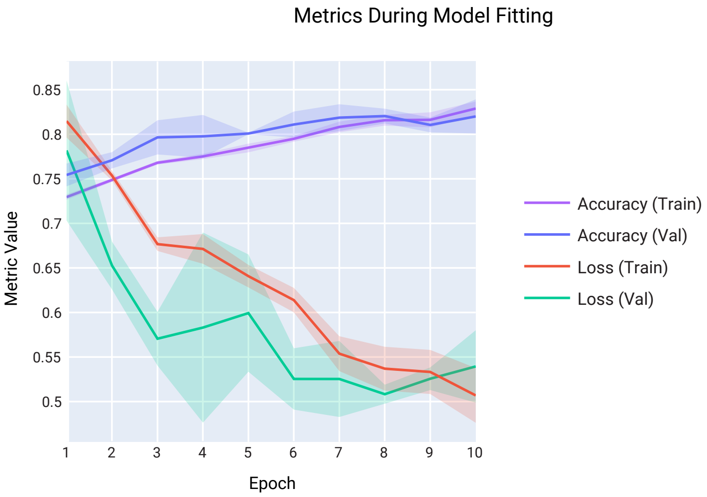


**Figure 6.6:** Fitting process of the skin image classifier during model training. The graph presents aggregated performance metrics across epochs, illustrating the model's learning progression with categorical accuracy and loss for both training and validation sets.

Accuracy ($acc$) is defined as the ratio of the sum of $TP$ (=True Positives) and $TN$ (=True Negatives) to the total number of cases examined.

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

where $FP$ are False Positives, and $FN$ False Negatives. This metric quantifies the proportion of correct predictions in relation to the total number of model predictions and is suitable for measuring during the training progress but not for evaluating model performance.

Figure 6.6 illustrates the training process of the model, with the training and validation metrics plotted across epochs. While the lines correspond to the average value of the respective metric over three runs, the shadow reveals the calculated standard deviation. Notably, the average sweet point across can be identified between epochs 9 and 10, indicating that the model reached a balance between learning and generalization before becoming over-fitted.

In-memory augmentation provided by TF (Martín Abadi *et al.* 2015), the ML framework used, was employed. This augmentation, executed on-the-fly while loading image data into the model, expanded the training data using the strategies „Rotation", „Shift", „Zoom", and „Flip". The Adam optimizer (Kingma *et al.* 2015) was used with Categorical Crossentropy (cf. 2.3) as loss function. An early stopping function utilizing the validation dataset was implemented to end the model training prematurely if there was no improvement in prediction performance on the validation data. The trained model files were stored in this stage, enabling them to be consumed as artifacts in subsequent stages.

**Encoding.** A separate code module calculates similarity vectors of samples within the *Encoding* stage in the ML pipeline, which is placed within the DAG after the model training and before the query stages. Various methods exist for determining essential similarity values in image files, including generating image representations by activating the last fully connected layer of the trained model. The resulting representations are then transformed into vectors, allowing for comparison based on their Euclidean distance (Sener *et al.* 2018).

This approach is further used by Beluch *et al.* 2018 to implement their density-based QS. Separating this stage from the QS stage offers two advantages: Firstly, the calculation of vectors is performed only as required, i.e., in ML pipeline configurations where the density-based QS is specified. Secondly, the determined vectors are stored as tracked data artifacts and made available to the query code module.

**Query.** To ensure an evaluation of different QSs, three QSs were implemented in the skin image analysis use case: Random serves as a baseline, while the other two represent prominent approaches - one information-based and the other representation-based:

- The *Random* based sampling strategy selects samples without following a specific pattern, thus providing a basis for benchmarking with other strategies.
- *Uncertainty* based sampling, which belongs to the information-based QS, selects samples where the model exhibits the highest uncertainty value in predictions, mainly focusing on those instances where the prediction is the „least confident" (Lewis *et al.* 1994a, c.f. 2.1.3).
- Graph *density* based sampling, proposed by Ebert *et al.* 2012, has been implemented as a representation-based QS. As previously described, this strategy focuses on the underlying data distribution and chooses samples, ideally avoiding repetitive dense regions, to achieve a diverse selection.

The parameterized size of the desired query set allows being specified in the ML pipeline configuration, following LIFEDATA's project template, just like the respective QS. As an output, this stage produces a data artifact in the form of a CSV file, which contains entries of image_ids, the query set of samples that the applied QS has selected.

**Validation.** In the validation stage, various model interpretation algorithms were implemented in the use case of skin image analysis. The results are presented in detail in Chapter 7.

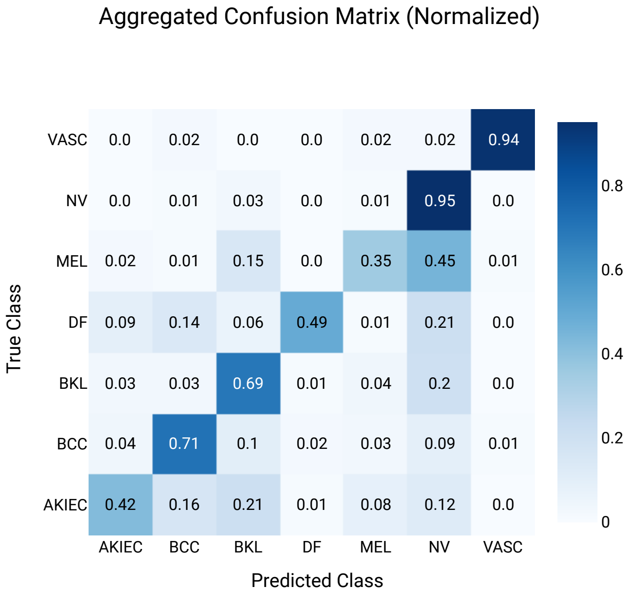


**Figure 6.7:** Confusion matrix of the trained skin image classifier. The heatmap depicts the normalized aggregated values, providing an analysis of the model's predictive accuracy on the test set for each class, with darker shades representing higher probabilities of class predictions.

**Table 6.3:** Aggregated model performance metrics of the skin image classifier. The table showcases the Precision, Recall, $F_1$-Score, and sample count for each class, with error margins of three runs.

| | Precision | Recall | $F_1$-Score | # Samples |
|---|---|---|---|---|
| AKIEC | 0.59 ± 0.13 | 0.42 ± 0.06 | 0.49 ± 0.04 | 62 |
| BCC | 0.71 ± 0.1 | 0.71 ± 0.1 | 0.71 ± 0.07 | 99 |
| BKL | 0.61 ± 0.11 | 0.69 ± 0.05 | 0.64 ± 0.05 | 204 |
| DF | 0.53 ± 0.11 | 0.49 ± 0.22 | 0.50 ± 0.16 | 24 |
| MEL | 0.68 ± 0.08 | 0.35 ± 0.05 | 0.46 ± 0.06 | 226 |
| NV | 0.89 ± 0.00 | 0.95 ± 0.02 | 0.92 ± 0.01 | 1355 |
| VASC | 0.86 ± 0.14 | 0.94 ± 0.05 | 0.90 ± 0.1 | 27 |
| micro avg | 0.83 ± 0.03 | 0.83 ± 0.04 | 0.83 ± 0.03 | 1997 |
| macro avg | 0.70 ± 0.05 | 0.65 ± 0.07 | 0.66 ± 0.06 | 1997 |
| weighted avg | 0.81 ± 0.02 | 0.82 ± 0.02 | 0.81 ± 0.02 | 1997 |

Precision, denoted as $P$, is the ratio of true positive predictions to total predicted positives:

$$P = \frac{TP}{TP + FP}$$

Recall, $R$, quantifies the proportion of actual positives correctly identified:

$$R = \frac{TP}{TP + FN}$$

The $F_\beta$-Score, a weighted hamornic mean of $P$ and $R$, is given by:

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R}$$

where $\beta$ is chosen based on the relative importance of $P$ versus $R$. In the case of $\beta = 1$, these are scored as equal.
Different averages can be calculated for the resulting metrics $M = \{P, R, F_\beta\}$. While micro determines the respective metric $M(y, \hat{y})$ globally based on the given ground truths set $y$ and set of the model's predictions $\hat{y}$, the macro average is calculated independently for each label $L$, resulting in a score that does not take the class frequency into account:

$$\text{macro-avg} = \frac{1}{|L|} \sum_{l \in L} M(y_l, \hat{y}_l)$$

Conversely, the weighted (wt) average compensates for label imbalance by scaling each label's metric relative to its prevalence:

$$\text{wt-avg} = \frac{1}{\sum_{l \in L} |y_l|} \sum_{l \in L} |y_l| \cdot M(y_l, \hat{y}_l)$$

**Evaluation.** The logic for evaluating the trained classifier is implemented in one of the final stages of the proposed ML pipeline. This step assesses the model's predictive performance on test samples, which were previously set aside during the *Data Splitting* stage. To ensure a robust performance evaluation, in this setting the entire ML pipeline was executed three times with different random seeds. The outcomes of these iterations were then aggregated to mitigate the variability induced by the training process's stochastic nature and provide a more reliable assessment of the model's performance.

The aggregated results are visualized in Figure 6.7. The normalized aggregated confusion matrix provides a detailed visual representation of the model's classification accuracy across different classes. The darker cells along the matrix's diagonal indicate higher predictive accuracy for the respective classes. For instance, the high values in the cells corresponding to 'NV' and 'VASC' classes imply a strong true positive rate, while off-diagonal cells highlight instances of misclassification. The matrix reveals that the model is particularly adept at predicting the 'NV' class, with a normalized value of 0.95, but shows a higher misclassification between the 'MEL' and 'NV' classes, as evidenced by the off-diagonal entries.

Complementing the confusion matrix, Table 6.3 summarizes precision, recall, $F_1$-Score, and support for each class, alongside macro and weighted averages. The precision column reflects the classifier's ability to not mislabel a sample as belonging to a particular class, while the recall indicates the model's proficiency in identifying all samples of a class. The $F_1$-Score offers a balance between precision and recall. The '# samples' column reports the number of images for each class in the test set, which contributes to understanding the weight of each class in the performance metrics. For example, the class 'NV' stands out with high precision and recall, denoting a high classification performance for this category. The macro and weighted averages provide an overall performance snapshot, with the macro average treating all classes equally and the weighted average taking into account the support for each class, which is particularly useful when class imbalances are present.

Based on these results, the AL part will now be examined in more detail.

### 6.1.3 Active Learning Simulation

This section evaluates the effort required for annotation, which is a crucial factor in enhancing the efficiency of incorporating human feedback into the AL methodology. The analysis focuses on whether the use of AL in annotating skin lesion images can reduce the required annotation effort and identifies which of the implemented QSs is most effective in preferentially selecting samples of rarer classes over those of the majority class during the labeling process.

To explore these aspects, experimental simulations of the labeling process were conducted. As previously discussed in the explanation of the ML pipeline, this was achieved using a specially developed code module within the Experiment Utilities designed for LIFEDATA, which provides annotations in each AL iteration for the samples requested by the QS. This approach is a common practice in the simulation of AL projects without using a real human oracle (Ghai *et al.* 2021). It is important to note that the laboratory conditions always ensure correct annotations and do not consider user errors.

The foundation for this was the implementation of the ML pipeline described in Section 6.1.2. Following the methodology proposed in Section 4.3.3, three sub-experiments were initialized in different experiment branches. Each sub-experiment simulated the labeling process, assuming that one of the implemented QSs, with a configured query set size of 400, is active. To validate the results, each sub-experiment was conducted with three different random seeds. For the results of the three runs, the average and standard deviation were calculated.

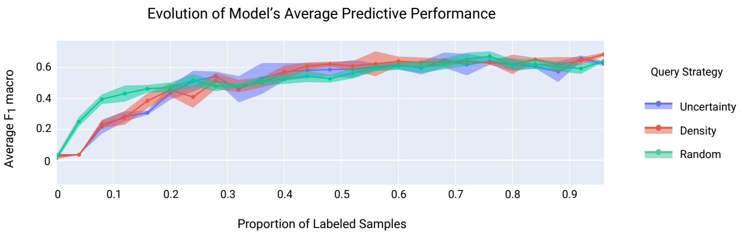


**Figure 6.8:** Average classification performance of the skin image classifier.

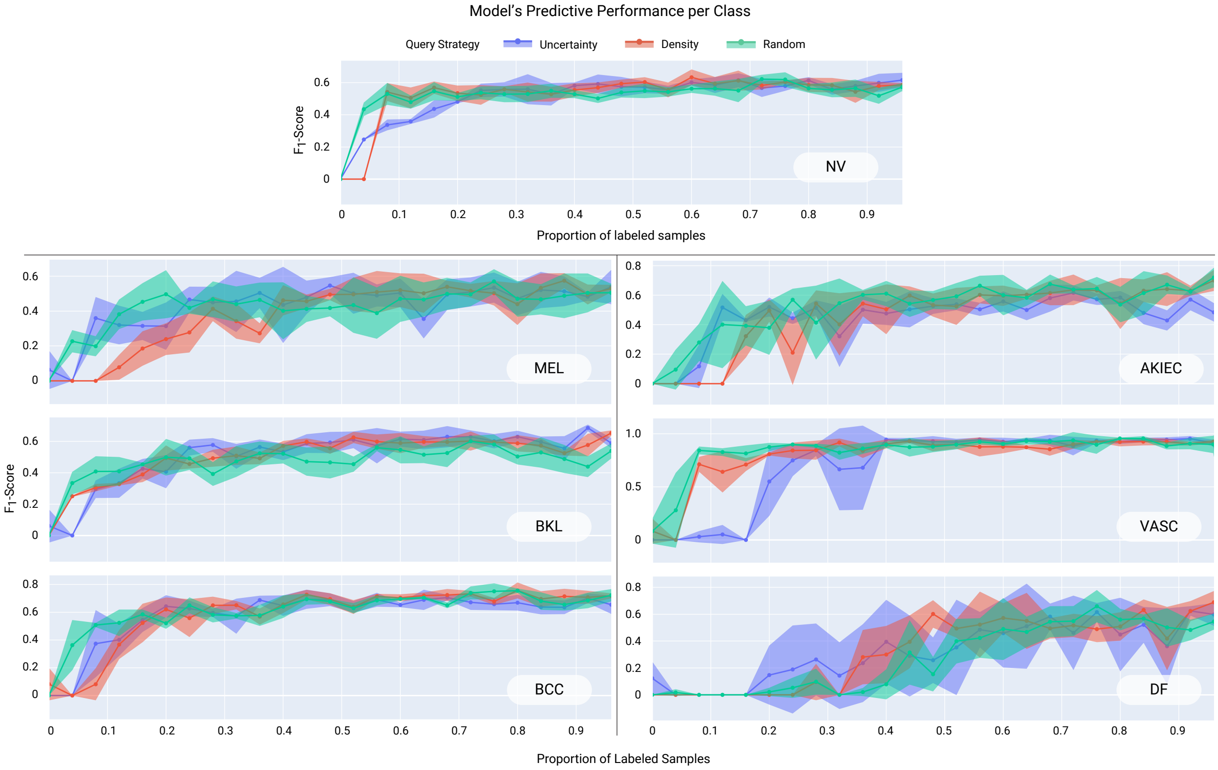


**Figure 6.9:** Skin image classification performance per class. The overview of the $F_1$-Scores achieved by the classifier for the respective classes in the annotation process illustrates the impact of different QSs on the model's predictive performance during the AL simulation.

Using the metadata collected in the MLflow instance, the analysis begins with examining the predictive performance of the trained model throughout the labeling process. For this simulation, the $F_1$-Score[3] serves as a simplified performance metric. Figure 6.8 illustrates the average $F_1$-Score across all classes, plotted against the number of annotated images. The colored lines represent the averaged value over the three experiments for each QS, while the shadows indicate the standard deviation. The figure shows that in the second AL iteration when less than 10% of the training data were annotated, the average $F_1$-Score for the experiments with the uncertainty and density-based QSs was around 0.2. In contrast, for the experiment with the random-based strategy (green line), it was approximately 0.4 across all classes.

Further evident in Figure 6.8, the $F_1$-Score increases for all QSs as the number of labeled training data grows. However, the uncertainty and density-based QSs lag behind the random-based sampling until the sixth AL iteration. In the simulation with the random-based QS activated, the peak in model predictive performance (0.67) was reached with about 75% of the training data annotated. The $F_1$-Scores in experiments using uncertainty- (0.66) and density-based (0.68) QSs only reached their maximum after 90% of the training data were annotated. However, it is observed that the predictive performance of the model, as measured by the $F_1$-Score, only marginally increases in all three experiments once 60% of the training data are annotated. Furthermore, none of the QSs leads to a significantly different predictive performance of the model after the conclusion of the simulations.

A more detailed investigation is carried out in Figure 6.9, which displays

[3] Unless otherwise stated, *macro* was set for the calculation of average values, where unweighted mean is calculated without taking the class imbalance into account.

the model's predictive performance for each class. The plots have a similar structure, whereby the performance of the classifier for the respective class is plotted on the ordinate, which is scaled differently in all subdiagrams. The nearly maximal predictive performance of the model for the majority class (NV) is already achieved with 20% of the training samples annotated, with the performance curve rising most steeply in the experiment with the activated random-based QS. Therefore, no significant advantage of a QS can be determined for this class.

However, examining the rarest class (DF) of the HAM10000 dataset reveals that the uncertainty and density-based QSs result in a significantly higher predictive performance of the model for this class in the AL iterations during the period from 20% to 60% of annotated training data, compared to the experiment with the random-based selection strategy. However, the high standard deviation should be noted, possibly due to the small number of training samples for this class.

In summary, none of the implemented QSs in this application scenario led to a significant difference in the model's predictive performance. Compared to the random sample selection benchmark, neither the uncertainty-based nor the density-based QS contributed to a meaningful reduction in annotation effort.

An evaluation of the query sets generated in the AL iterations is conducted to analyze the implemented QSs in the context of the selected classes and their frequency distribution. This is made possible by the established project setup and the integration of the data versioning system in LIFEDATA, with the artifacts managed by DVC from the data remote of the AL project being used for targeted analyses.

Figure 6.10 provides an overview of the distribution of classes within the generated query sets. To increase comparability, these are once again placed in relation to the proportion of the labeled training data quantity depicted on the x-axis. The individual classes are identifiable in the subplots, where the ordinate show the quantity of samples selected by the QSs in absolute numbers. Similarly, the data is represented using lines, with the calculated means across the three random seeds and the shading indicating the standard deviations.

In the evaluation, the class selection of the three implemented QSs is compared, with the random-based QS serving as a benchmark once again. The relatively uniform curve trajectories (green line) illustrate an expected pattern: as the selection is random, approximately the same number of samples from each class are selected throughout the annotation process. The respective absolute count reflects the proportion of class representation in the HAM10000 dataset, in which, as a reminder from Figure 8.2, the majority class (NV) occurs approximately 60 times more frequently than the rarest class (DF).

For the Uncertainty-based (blue line) and Density-based (red line) QSs, an initially unusually high selection of the majority class (NV) is observed in the class selection, which rapidly decreases during the simulated annotation process. The analysis of samples from other classes shows that in the first half of the annotation process, an over-proportional number of samples are queried for annotation. This suggests that the model generates more uncertain prediction values for these classes during this phase.

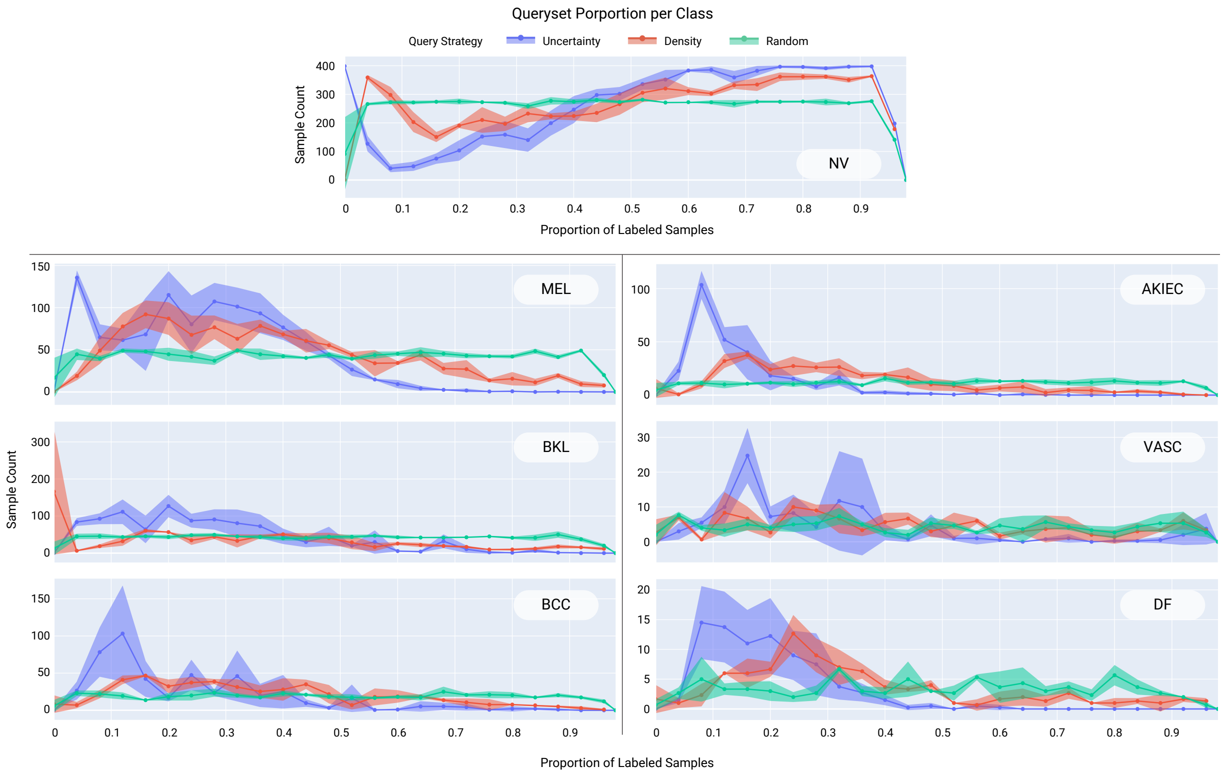


**Figure 6.10:** Query set proportion per class. The subplots showcasing the comparison of implemented QSs during the annotaion simulation. The graph shows the impact of Uncertainty- and density-based strategies in selecting skin lesion images from rare classes, demonstrating their advantages over random selection.

Only in the second half of the annotation process does the number of samples from the majority class (NV) increase, while the curves for the other classes tend towards zero. This suggests that the samples from the rarer classes were almost fully annotated at this point, and the remaining samples mostly belonged to the majority class, as evidenced by reaching the maximum value of 400, the chosen size of the query set.

In summary, the implemented selection strategies – Uncertainty-based and Density-based – are equally effective in preferentially selecting skin lesion images from rare classes. They offer a significant advantage over random selection, thereby enhancing the targeted acquisition of human feedback.

## 6.2 ECG Signal Classification

Cardiovascular diseases remain the leading cause of global mortality, with heart conditions such as arrhythmias posing significant health challenges (Roth *et al.* 2020). In this context, ECG analysis emerges as a critical diagnostic tool, enabling the detection of irregular heart rhythms that may indicate underlying heart conditions. The advent of AI-supported ECG signal classification has sparked considerable interest among researchers, evidenced by initiatives like the PhysioNet Computing in Cardiology Challenges (Goldberger *et al.* 2000; Alday *et al.* 2020; Reyna *et al.* 2021), which have showcased the potential of AI in enhancing ECG analysis through ML techniques.

ECG signals, which represent the heart's electrical activity, require intricate analysis, often involving steps such as noise reduction, feature extraction, and the segmentation of relevant signal components, e.g., identifying the QRS complex, a part of the characteristic ECG curve representing the electrical activity of the heart's ventricular muscles. The subsequent classification of these signals into normal or various arrhythmic categories is a critical step for diagnostic support. While traditional approaches to ECG signal classification have relied on manual feature extraction, recent strides in AI research have demonstrated the superiority of DNNs, in automating feature extraction and classification with higher accuracy (Acharya *et al.* 2017; Bizopoulos *et al.* 2019; Liu *et al.* 2021). These models have not only matched but in some cases exceeded the performance of cardiac specialists in identifying arrhythmias (Rajpurkar *et al.* 2017; Hannun *et al.* 2019), underscoring the potential of CDSSs in this field.

However, the development of DL-based ECG signal classifiers face challenges, including the shortage of large, annotated ECG signal datasets necessary for supervised training of these models. The following case study on ECG signal classification aims to explore the AL workflow in this context using LIFEDATA.

By applying LIFEDATA to this use case, it will be re-examined whether the annotation effort for the training data of a 12-lead ECG classifier can be optimized through AL, thereby increasing the efficiency of data labeling. Furthermore, this use case was conducted with real humans in the loop, meaning that a web interface was provided for annotating the training data, where medical professionals could provide their feedback. The setup of this real-world application offers the potential to assess the efficiency of data labeling enabled by the LIFEDATA framework and evaluate the traceability of labels it facilitates. Initially, the following sections will discuss the data foundation and project implementation.

### 6.2.1 Data Basis

The significant attention the research community has devoted to analyzing ECG data has led to the availability of a comprehensive number of public datasets containing diverse ECG signals. These datasets serve various purposes, including long-term ECG, exercise ECG, and recordings using single lead or multi-channel systems.

**Table 6.4:** Relevant data sources with ECG signal data. The given number of samples corresponds to the number of records that meet the requirements for signal length and resolution as well as belong to one of the observed classes.

| Data Source | Description | # Samples |
|---|---|---|
| Chap.Shaoxing | Chapman University & Shaoxing People's Hospital (Zheng *et al.* 2020b) | 9,873 |
| CPSC2018 | China Physiological Signal Challenge 2018 (Liu *et al.* 2018) | 3,148 |
| G12EC | Georgia 12-lead ECG Challenge | 9,266 |
| Ningbo | Ningbo First Hospital (Zheng *et al.* 2020a) | 34,391 |
| PTB_XL | Physikalisch Technische Bundesanstalt (Wagner *et al.* 2020) | 21,045 |

**Table 6.5:** Available metadata of the ECG records..

| Variable | Description |
|---|---|
| sample_id | Unique identifier for each record |
| channel (I-V6) | Channel description, including min&max values, resolution and sampling rate |
| dx | The diagnosis, described in Table 6.6 |
| age | The age of the patient |
| sex | Information on the gender of the patient |

In the context of demonstrating the LIFEDATA framework for the ECG signal classification use case, a careful selection of datasets containing signals from various sources was made. This selection was strictly limited to samples featuring 12-lead ECG recordings with a minimum duration of 10 seconds of resting ECGs. Datasets comprising 12-lead resting ECG samples exceeding a duration of 10 seconds were excluded. Furthermore, the selection was confined to raw signals with a sampling frequency of at least 500 $Hz$ and a resolution of at least 16 bits.

The utilized data sources are detailed in Table 6.4, including the number of corresponding samples. Due to the availability in WFDB format (Goldberger *et al.* 2000), additional metadata are available, described in Table 6.5, which includes information such as the designations of the twelve channels, the sampling rate, and the range of signal values. Further provided are the diagnostic labels assigned by cardiologists, the age, and the gender of the patients.

The most common diagnostic classes to be predicted by the ECG signal classifier during the course of the use case were defined. The selection of these classes is presented in Table 6.6, including the abbreviations and

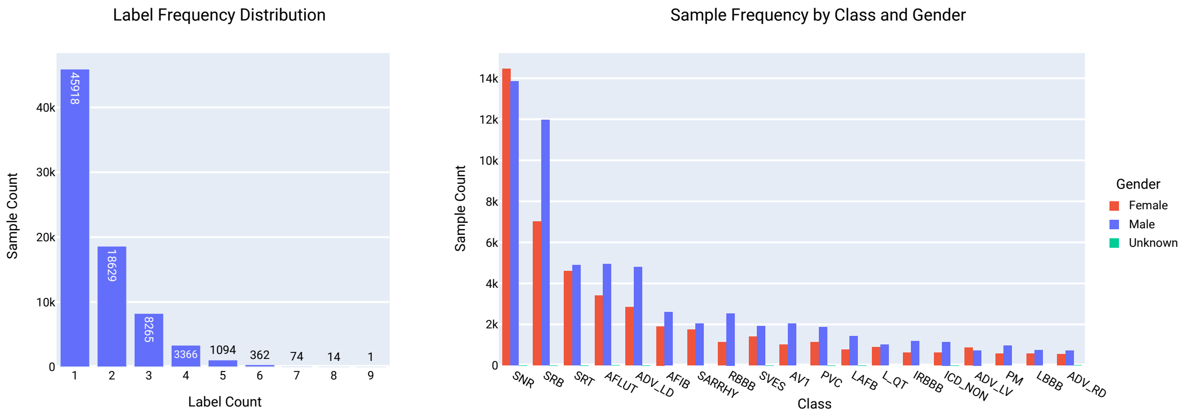
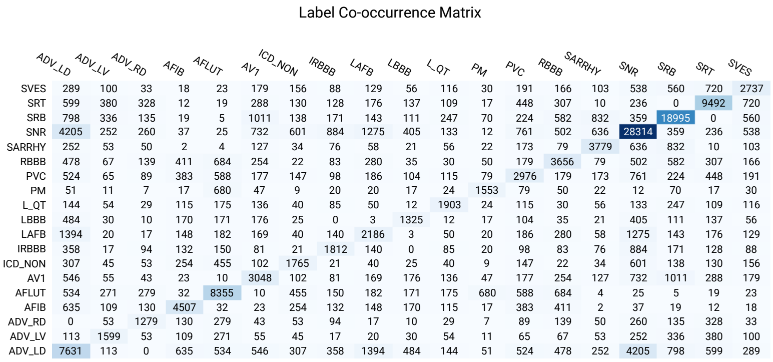


**Figure 6.11:** Overview of sample and label frequencies. The occurrence of one or two labels per sample is most common, but there are samples with up to nine labels. The chart of sample frequency is sorted in descending order, with 'SNR' being the most common and 'ADV_RD' the rarest class in the analyzed dataset.

| Class | Description |
|---|---|
| ADV_LD | Advanced Leftward Deviation |
| ADV_LV | Advanced Left Ventricular Abnormalities |
| ADV_RD | Advanced Rightward Deviation |
| AFIB | Atrial Fibrillation |
| AFLUT | Atrial Flutter |
| AV1 | First-degree Atrioventricular Block |
| ICD_NON | Non-specific Ischemic |
| IRBBB | Incomplete Right Bundle Branch Block |
| LAFB | Left Anterior Fascicular Block |
| LBBB | Left Bundle Branch Block |
| L_QT | Long QT Interval |
| PM | Pacemaker Present |
| PVC | Premature Ventricular Contractions |
| RBBB | Right Bundle Branch Block |
| SARRHY | Sinus Arrhythmia |
| SNR | Sinus Rhythm |
| SRB | Sinus Rhythm with Bradycardia |
| SRT | Sinus Rhythm with Tachycardia |
| SVES | Supraventricular Ectopic Beats |

**Table 6.6:** Class description of observed 12-lead ECG signal samples within several datasets of the PhysioNet Challenge 2021 hosted by Reyna *et al.* 2021. The class description modified in reference to Wagner *et al.* 2020.



Label Co-occurrence Matrix

| | ADV_LD | ADV_LV | ADV_RD | AFIB | AFLUT | AV1 | ICD_NON | IRBBB | LAFB | LBBB | L_QT | PM | PVC | RBBB | SARRHY | SNR | SRB | SRT | SVES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SVES | 289 | 100 | 33 | 18 | 23 | 179 | 156 | 88 | 129 | 56 | 116 | 30 | 191 | 166 | 103 | 538 | 560 | 720 | 2737 |
| SRT | 599 | 380 | 328 | 12 | 19 | 288 | 130 | 128 | 176 | 137 | 109 | 17 | 448 | 307 | 10 | 236 | 0 | 9492 | 720 |
| SRB | 798 | 336 | 135 | 19 | 5 | 1011 | 138 | 171 | 143 | 111 | 247 | 70 | 224 | 582 | 832 | 359 | 18995 | 0 | 560 |
| SNR | 4205 | 252 | 260 | 37 | 25 | 732 | 601 | 884 | 1275 | 405 | 133 | 12 | 761 | 502 | 636 | 28314 | 359 | 236 | 538 |
| SARRHY | 252 | 53 | 50 | 2 | 4 | 127 | 34 | 76 | 58 | 21 | 56 | 22 | 173 | 79 | 3779 | 636 | 832 | 10 | 103 |
| RBBB | 478 | 67 | 139 | 411 | 684 | 254 | 22 | 83 | 280 | 35 | 30 | 50 | 179 | 3656 | 79 | 502 | 582 | 307 | 166 |
| PVC | 524 | 65 | 89 | 383 | 588 | 177 | 147 | 98 | 186 | 104 | 115 | 79 | 2976 | 179 | 173 | 761 | 224 | 448 | 191 |
| PM | 51 | 11 | 7 | 17 | 680 | 47 | 9 | 20 | 20 | 17 | 24 | 1553 | 79 | 50 | 22 | 12 | 70 | 17 | 30 |
| L_QT | 144 | 54 | 29 | 115 | 175 | 136 | 40 | 85 | 50 | 12 | 1903 | 24 | 115 | 30 | 56 | 133 | 247 | 109 | 116 |
| LBBB | 484 | 30 | 10 | 170 | 171 | 176 | 25 | 0 | 3 | 1325 | 12 | 17 | 104 | 35 | 21 | 405 | 111 | 137 | 56 |
| LAFB | 1394 | 20 | 17 | 148 | 182 | 169 | 40 | 140 | 2186 | 3 | 50 | 20 | 186 | 280 | 58 | 1275 | 143 | 176 | 129 |
| IRBBB | 358 | 17 | 94 | 132 | 150 | 81 | 21 | 1812 | 140 | 0 | 85 | 20 | 98 | 83 | 76 | 884 | 171 | 128 | 88 |
| ICD_NON | 307 | 45 | 53 | 254 | 455 | 102 | 1765 | 21 | 40 | 25 | 40 | 9 | 147 | 22 | 34 | 601 | 138 | 130 | 156 |
| AV1 | 546 | 55 | 43 | 23 | 10 | 3048 | 102 | 81 | 169 | 176 | 136 | 47 | 177 | 254 | 127 | 732 | 1011 | 288 | 179 |
| AFLUT | 534 | 271 | 279 | 32 | 8355 | 10 | 455 | 150 | 182 | 171 | 175 | 680 | 588 | 684 | 4 | 25 | 5 | 19 | 23 |
| AFIB | 635 | 109 | 130 | 4507 | 32 | 23 | 254 | 132 | 148 | 170 | 115 | 17 | 383 | 411 | 2 | 37 | 19 | 12 | 18 |
| ADV_RD | 0 | 53 | 1279 | 130 | 279 | 43 | 53 | 94 | 17 | 10 | 29 | 7 | 89 | 139 | 50 | 260 | 135 | 328 | 33 |
| ADV_LV | 113 | 1599 | 53 | 109 | 271 | 55 | 45 | 17 | 20 | 30 | 54 | 11 | 65 | 67 | 53 | 252 | 336 | 380 | 100 |
| ADV_LD | 7631 | 113 | 0 | 635 | 534 | 546 | 307 | 358 | 1394 | 484 | 144 | 51 | 524 | 478 | 252 | 4205 | 798 | 599 | 289 |

**Figure 6.12:** Label co-occurrence matrix. It shows which label occurs in combination with which other labels. The coloring illustrates that samples labeled with 'SNR' are often exclusively labeled with 'SNR'.

their meanings. Figure 6.11 displays the frequency distribution of the classes, starting with the most common class, in descending order. The distribution indicates a more balanced ratio to the more frequent classes, although rarer classes are significantly underrepresented, leading to an imbalance. This is particularly relevant in the context of investigating QSs.

A distinctive characteristic of this use case is its characterization as a multi-label problem, where an ECG sample can have multiple true labels. The distribution of the number of labels per sample is depicted in Figure 6.11, with a dominance of samples with a single label, followed by samples with two labels. Samples with three or more labels are present, with their class combinations shown in the label co-occurrence matrix in Figure 6.12.

The analysis of the patient population, as shown in Figure 6.13, shows a balanced gender ratio. The median age is indicated as (male (m): 63, female (f):61, unknown (u):68), the lower quartile as (m:51, f:48, u:64), and the upper quartile as (m:71, f:72, u:72), suggesting an overrepresentation of patients within this age range. The sample frequency in relation to demographic information shows a higher incidence of male patients. Information on ethnic background is not included in the metadata.

As was evident in the skin image analysis use case (cf. 6.1), these limita-

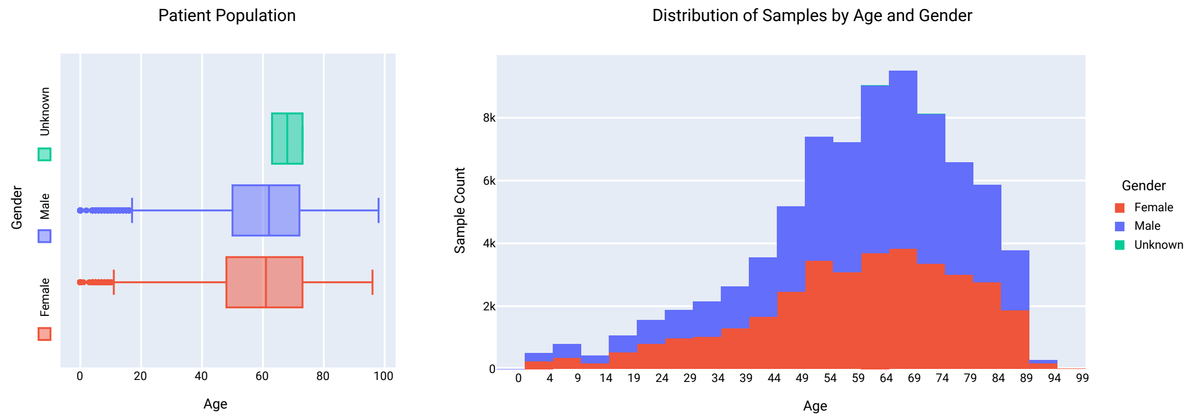


**Figure 6.13:** Distribution of samples in relation to patient population. While the demographic characteristics appear balanced, the histogram of sample count, including gender information, reveals an overrepresentation of male patients.

tions of the data basis must be considered in the generalizability of the results. However, since the primary goal of applying LIFEDATA is not on the outcomes of the ML pipeline but on exploring the applicability of AL and aspects of data annotation, these publicly accessible data sources are suitable. Firstly, annotations already exist for the data, a significant portion of which have undergone a quality assessment. Secondly, the data have proven to be a valuable training basis for numerous research works, such as during the Physionet Challenges 2020 (Alday *et al.* 2020) and 2021 (Reyna *et al.* 2021).

### 6.2.2 Application

The following section illustrates the adjustments to LIFEDATA for the use case of the 12-lead ECG signal classifier. In addition to presenting the project setup, a detailed overview of the ML pipeline implementation is provided.

#### Project Setup

In the implementation of the use case, the LIFEDATA project template was enhanced with components that are foreseen in the LIFEDATA reference architecture, namely MLOps components. Additional web-based components specified by LIFEDATA, such as the dashboard UI, annotations UI, and the relational database for storing labels provided by a human oracle, were provisioned to implement a human-in-the-loop scenario. As a result, this project setup materialized an infrastructure that embodies the principles detailed in Section 4.2.3.

The architectural schema of this infrastructure is illustrated in Figure 6.14, elucidating the interplay and functionality of each component. For clarity, it is divided into three zones: Development, Machine Learning/-Experimentation, and Operations, each housing different infrastructure components.

As part of the Development Zone, a GitLab instance was employed as an SCM server, managing the code repositories of both the ETL project and the LIFEDATA AL project instance. Within these repositories, version

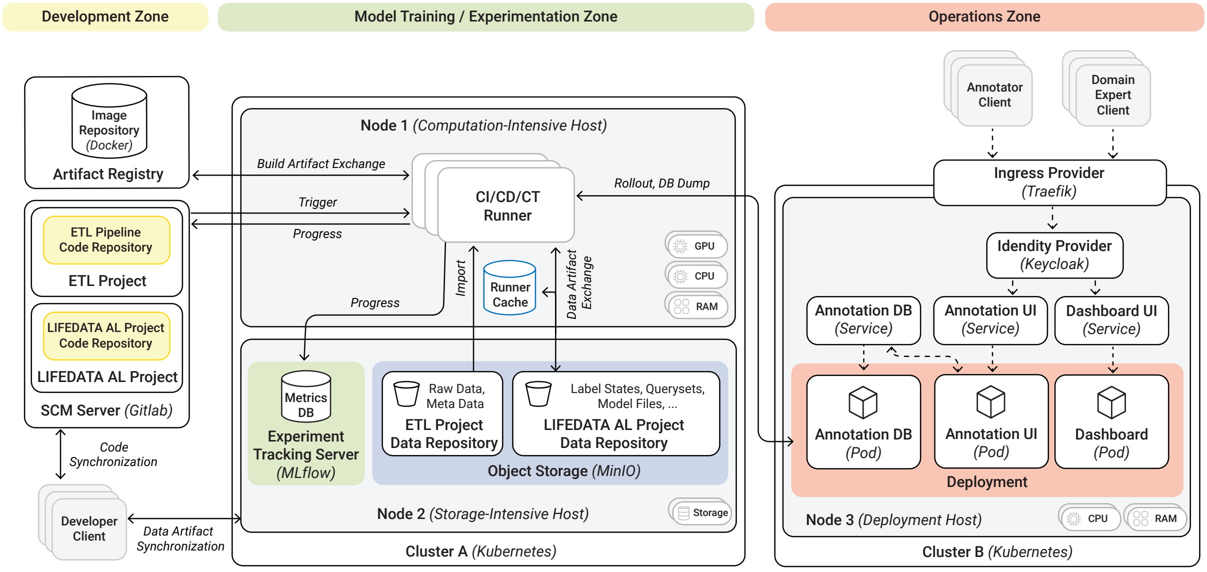


**Figure 6.14:** Project setup for the ECG signal classification use case. The figure illustrates a process and system perspective as a fusion between the components of the LIFEDATA reference architecture (Figure 5.2) and the applied infrastructure concept as described in Figure 4.3, showcasing the interaction of the individual elements within the specific LIFEDATA project instance. In this setup, the infrastructure components are managed within clusters to accommodate the increased demands of computation- and data-intensive processes. The architectural diagram illustrates that these are divided into a development zone, model training / experimentation zone and an operations zone. The deployment of the container-based components for the annotation and dashboard UI is housed within a separate cluster.

control is applied to all code modules and associated data placeholder files. Furthermore, these projects encompass the implementation of a CI/CD/CT component, which again establishes the instructions for the CI/CD/CT runner. One of the tasks is creating a Docker image, which is published as the build artifact to an image repository. This artifact forms the basis for subsequent tasks, such as running the ML pipeline within a containerized environment on the CI/CD/CT runner.

The CI/CD/CT runner is deployed on one of two nodes designated for computation-intensive tasks within Kubernetes Cluster A, equipped with the appropriate hardware specifications. This includes two Intel Xeon Silver 4110 CPUs, 377 GB of RAM, one TB of SSD storage, two NVIDIA Quadro P6000 GPUs, and two NVIDIA Quadro RTX 6000 GPUs, each with 24 GB of graphic memory. In this use case implementation, the containers of the CI/CD/CT runners are mounted with a shared directory used as a local cache, which DVC manages as content-addressable storage (Iterative 2024).

In addition to the elements above, the infrastructure hosts the data repositories for the respective projects on a storage-centric node within the same Kubernetes cluster. This node, equipped with 32 GB of RAM and 12 TB of SSD storage, provides hardware resources for storage-intensive services within the infrastructure. This includes the Experiment Tracking Service, which, for this use case, is again realized in an MLflow instance (Zaharia *et al.* 2018). The database provided by MLflow for storing metrics has been integrated into the code modules, making all collected metrics on the model and query set visible to developers, both in the experiments and the deployed production environment.

Analogous to the role of the SCM server as a Git remote for code artifacts, these storage locations are configured as remote entities within the data versioning application. The tasks of the CI/CD/CT module, associated with the execution of the ML pipeline, are explicitly defined to ensure that all resulting data artifacts, such as the trained model and the query set, are transmitted to the object storage server - realized through MinIO (MinIO Inc. 2024) - using the S3 protocol into the respective buckets post-execution. Similarly, this setup allows the import of artifacts (a specific version of the trained model or a query set) by various CI/CD/CT runners in cases where they are not present in the local cache and are needed for downstream jobs, such as deploying the deployment.

**Deployment.** As a third area in the operations zone of the infrastructure diagram, the provisioning mechanisms are realized. Alongside the automation of the training pipeline, the principles of continuous deployment are prominently featured. The dedicated Kubernetes Cluster B within this zone includes Node 3 (Deployment Host), designed to cover the provision of necessary operational components, including the relational database for storing annotations and web-based GUIs for annotations and the dashboard. These services are contained in Pods, a nomenclature native to Kubernetes (The Kubernetes Authors 2024a), and are orchestrated to provide functions for storing annotations made by annotators, as well as for presenting information in the dashboard for domain experts.

Traefik, as instance of an ingress provider, handles traffic ingress to these services, which adeptly routes requests to the internal services (Traefik Labls 2024). Following the LIFEDATA framework, a Keycloak (Keycloak Authors 2023) instance is deployed as the Identity Provider, managing users and access control to the respective services. This ensures that different front-end users (annotators and domain experts) can authenticate and that the integrity of their input, such as provieded labels, is ensured.

In this use case, the automated workflow defined within the CI/CD/CT component involves a scheduled job that runs the LIFEDATA Machine Learning Service nightly. The CI/CD/CT runner in the production environment, upon deploying the annotation UI Pod — which further encompasses the logic of the annotation widget - rolls out the query set and the preprocessed signal data of the contained sample IDs.

The deployment process begins with initiating a job within the CI/CD/CT component, which covers the execution of the ML pipeline in the production environment. Initially, a database dump from the active annotation DB is requested using the CLI logic implemented in the LIFEDATA core framework (see Section 5.3.1) containing the annotations. Upon receiving the new annotations, the Machine Learning Service is invoked, which includes the execution of the ML pipeline with the current annotation status, leading to the training of the model and the generation of a new query set. After the ML pipeline is executed, the artifacts, including the updated model and query set, are pushed to the object storage and, regarding the concept of 4.2.3, the updated data placeholder files were committed and pushed to the corresponding branch of the SCM server.

The deployment mechanism utilizes the inherent capabilities of Kubernetes, implementing a rolling update strategy to ensure that services remain available and are version-controlled throughout the deployment process (The Kubernetes Authors 2024b). This rolling update is executed by incrementally scheduling the Pod instances (annotation UI with the new query set and Dashboard UI with updated information on annotations and model metrics) on Node 3 within Cluster B.

**Dashboard.** The implementation of a real-world scenario involving annotations provided by humans necessitated the deployment of an enhanced dashboard component for domain experts, allowing for customizable analyses. Visualizations were created using Plotly (Plotly Technologies Inc. 2015), a low-code Python framework for web applications to achieve this. This framework enables the prepared and user-friendly display of model results and other relevant metrics within a web-based UI.

To ensure that the information visible to the user accurately reflects the specific state of the data and model, such as the daily status, the Dashboard Pod, which hosts the Dashboard UI, incorporates functions that allow for direct integration with the version control systems.

Two main concepts underpin the implementation to achieve this integration:

- Version synchronization: Each time the dashboard pod is deployed or updated, the current or a user-specified commit hash from the Git repository is retrieved. This ensures that the displayed data and metrics are synchronized with the respective state of the codebase. Using the branching strategy described in Section 4.4, for instance, the productive AL branch or a commit from it, which contains the repository over the placeholder files tracked by the SCM can be specified.
- Remote Data Fetching: The functions implemented within the dashboard component are then capable of retrieving data managed by the data version control system from the object storage. This allows loading the chosen versions of data and model repositories that correspond to the commit hash and passing them to the implemented methods for data evaluation.

Consequently, the dashboard user has control and, therefore, full transparency over which version is checked out to determine results and conduct analyses at a selected point in time.

### Machine Learning Pipeline

The adaptation of the ML pipeline definition proposed in the project template was adapted for the 12-lead ECG signal classifier use case. However, the changes were more extensive and can be seen in Figure 6.15, which shows the resulting ML pipeline as DAG.

The step for generating pseudolabels was removed, as no semi-supervised scenario was implemented in this use case. An additional modification compared to the original ML pipeline definition by LIFEDATA is the changed order of the individual steps. Since the raw data does not fluctuate significantly in this use case, but the annotation status undergoes more significant dynamics during the realization of experiments for

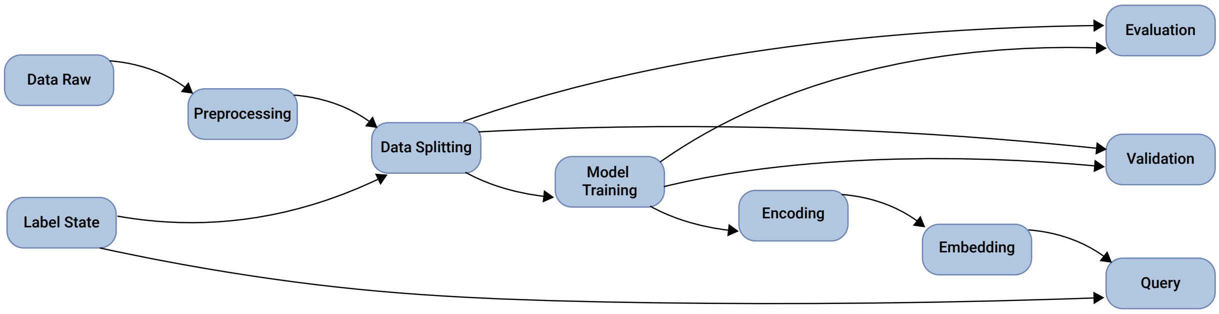


**Figure 6.15:** LIFEDATA's ML pipeline definition applied to the ECG signal classification use case. Again, the semi-supervised stage has been dropped, and the order of the initial pipeline was modified by moving the preprocessing stage, which is highly computationally intensive and less fluctuating in this use case, closer to the beginning of the ML pipeline. Additional adjustments were implemented in separate stages for generating vectors, used for representation-based QSs.

data labeling, the computationally and time-intensive preprocessing procedures were placed before the data splitting stage. This ensures that the pre-processed signal data is kept available for a frequent execution of the ML pipeline. Further changes included the insert of additional stages *Encoding* and *Embedding* for generating vecotrs, used for representation-based QSs. The specific implementations in each stage are presented below to provide a more detailed understanding of the ML pipeline.

**Data Raw.** As described in Section 6.2.1, multiple data sources with ECG signal data are utilized to realize this use case. Following the proposal of the LIFEDATA project template (see Section 5.4.2), the extraction of raw data from publicly accessible databases occurs in a separate ETL project. In this project, code modules with routines for filtering the relevant 12-lead ECG signals in their raw state by duration, class, and quality regarding resolution and sampling frequency are implemented.

For the remaining, for this use case selected, raw 12-lead ECG signals, a version (calculated hash value the ETL pipeline's output) is created according to the data repository concept, which is referenced as a placeholder file in the *Data Raw* stage of the implemented ML pipeline of the LIFEDATA AL project and is automatically imported using DVC during the execution of the ML pipeline.

**Label State.** According to the project template of LIFEDATA, in the *Label State* stage, index files describing the raw data's labeling state are being created, thus enabling the reproducibility of individual AL cycles. Contrary to the use case of skin image analysis (see Section 6.1), where the labeling process is exclusively simulated through a dedicated module within the experimental utilities, the annotation in the use case of the 12-lead ECG signal classifier occurs in a real scenario by human experts.

The components provided by LIFEDATA, including the web-based annotation UI and the relational database, integrate into the labeling process for storing annotations provided by human oracles. In the production environment, the index files created in the *Label State* stage are updated through a database dump of the relational database containing the annotations. Since this task, similar to the approach in experiments where the code module simulates the annotation process, occurs independently of the ML pipeline managed by the data versioning system, any change in the input data (new labels) leads to a recalculation of the subsequent stages of the ML pipeline that depend on the current labeling state. This ensures that subsequent artifacts (e.g., the trained model) constantly reference a managed and, therefore, reproducible labeling state.

**Preprocessing.** In the preprocessing stage of the ML pipeline tailored for 12-lead ECG use cases, the methodology deviates from logic, e.g., for image preprocessing. To enhance the quality of the raw signals, this stage includes the implementation of filtering and normalization processes adapted for signal data.

The application of a baseline filter aims to remove very low frequencies caused by physiological phenomena such as patient movements or respiratory effects. This step minimizes signal distortions that could obscure the medically relevant ECG signal. Using a notch filter eliminates artifacts in the raw signal that could be present due to power line frequencies, considering frequencies of 50, 60, 100, and 120 $Hz$. The application of a bandpass filter excludes frequencies outside defined ranges. This filtering focuses the analysis on relevant frequency bands and significantly reduces possible background noise.

Furthermore, the preprocessing stage optimizes the signal's amplitude values, adjusting the values after removing outliers based on the lower 2% and upper 98% percentiles. Additionally, consultations with cardiologists have determined that standardizing the sampling rate to 400 $Hz$ is feasible, resulting in 4,000 values for a 10-second signal length. This resolution provides a balanced ratio between capturing detailed signal characteristics and reducing the input data volume for model training. The resulting preprocessed signals are pushed as output to object storage and used both for display in the annotation UI and as input for downstream stages of the ML pipeline.

**Data Splitting.** Implementing the data splitting stage encompasses two splitting strategies to ensure reliable evaluation of QSs and models. During the execution of experiments (simulated annotation process), the training dataset is sampled proportionally as follows: 70% of the samples are used for training the model. For validation, 10% of the data are utilized during the training process in the *Model Training* phase to prevent potential overfitting through early stopping mechanisms. The remaining 20% of the data are reserved as a test dataset for measuring the model's performance.

For the production environment, the logic of the static test set provided in the LIFEDATA project template is employed. This involves defining the sample IDs through a separate index file, which is statically defined and does not change and therefore ensures that these IDs are consistently allocated to the test pool in each execution of the ML pipeline, regardless of the rest of the configuration. The size of the test dataset in production is set to 20% of the total data. From the remaining 80% of annotated data, a proportional split of 10% for validating model performance during the training process is conducted in each execution of the ML pipeline. The calculation of model metrics at the end of the model (re-)training is performed on the static test dataset, allowing for a consistent comparison of models and QSs across annotation iterations. In both splitting strategies, index files are created as the output of this stage, which indicate the assignment of the samples to the train, test and validation datasets.

**Model Training.** In this stage of the ML pipeline, a code module loads the data into the model, and the algorithm is adjusted accordingly. The model architecture implemented is a combination of a CNN and RNN for predicting ECG signal data, as proposed by Xu *et al.* 2020. This model architecture consists of two convolutional layers with 5x5 kernels and ReLU activations, followed by four residual blocks, two bidirectional short-term storage layers (biLSTM), and two fully connected layers (Xu *et al.* 2020).

The settings for the hyperparameters batch size, number of epochs, and learning rate were parameterized, and optimization on the utilized data basis yielded an optimal configuration of `BATCH_SIZE=32` and an initial `LEARNING_RATE=0.00 08`. The NAdam optimizer (Dozat 2015) was implemented, and the Binary Cross Entropy loss function was used.

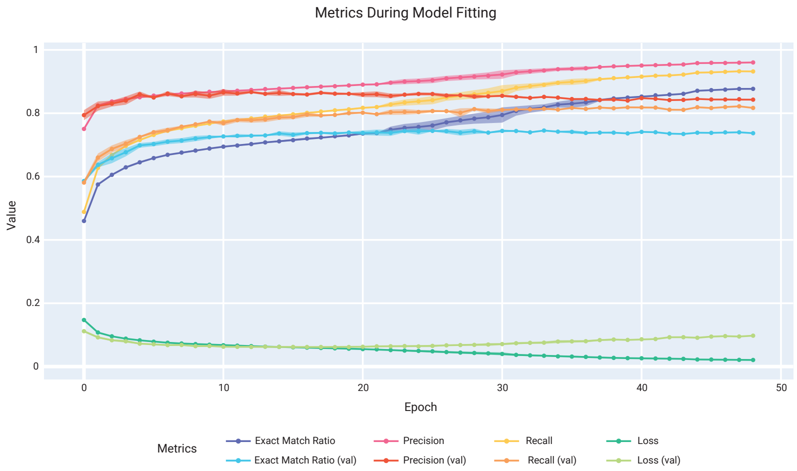


**Figure 6.16:** Fitting process of the ECG signal classifier during model training. The graph presents aggregated performance metrics across epochs, illustrating the model's learning progression with various metrics for both training and validation sets.

The accuracy score can be calculated as an indicator of the model prediction performance during model fitting (see 6.1.2), whereby the function returns the partial quantity accuracy in the multi-label classification problem. In the strictest case, this corresponds to the Exact Match Ratio ($EMR$), given by:

$$\text{EMR} = \frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{y}_i)$$

where $I(x)$ is the indicator function and results to 1 for a full match of all labels in a prediction $\hat{y}$ with the ground truth $y$, or 0 otherwise.

Figure 6.16 shows the model's fitting curve during the training phase. The plotted metrics correspond to the average value of three runs initialized with different random seeds. Again, the shadow reveals the calculated standard deviation. As seen in Figure 6.16, the average sweet point is reached around epoch 20. An early stopping function utilizing the validation dataset was implemented using the TF library to terminate the model training prematurely if the prediction performance on the validation data did not improve (Martín Abadi *et al.* 2015). The value of `EPOCHS` was initially set to 50 in each ML pipeline execution, whereby the early stopping function could halt the training earlier if necessary.

Similarly, the learning-rate scheduler provided by TF, which dynamically adjusts the learning rate between epochs, was implemented. The „ReduceLROnPlateau" strategy, where the learning rate is reduced once a defined metric fails to improve, was chosen.

The trained model file was saved during this phase to be used as an artifact in subsequent stages.

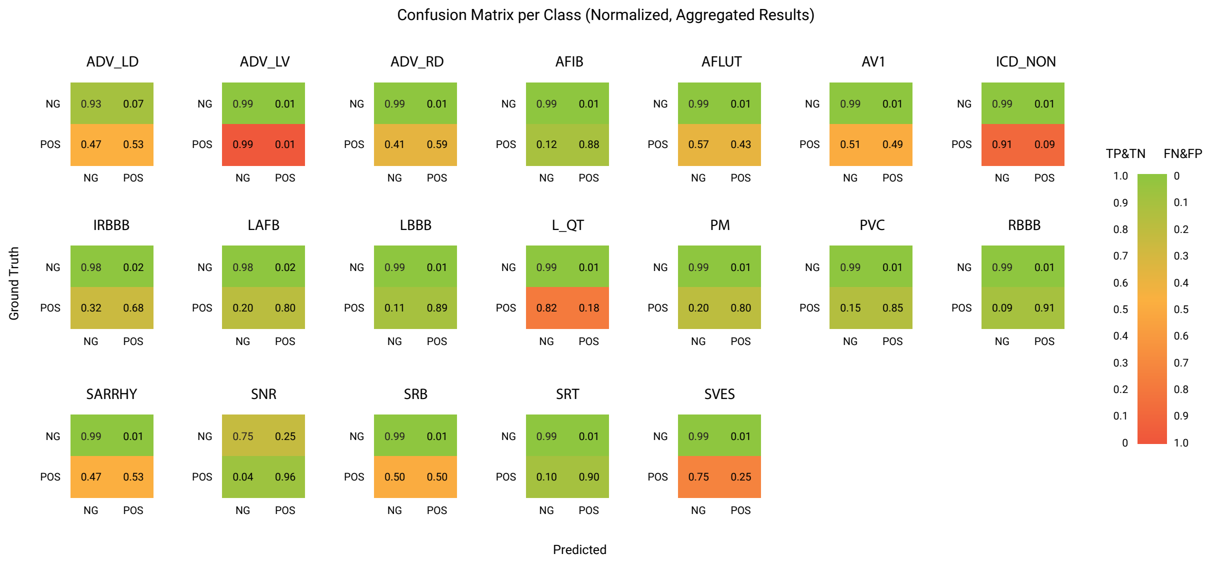


**Figure 6.17:** Confusion matrix of the trained ECG signal classifier. Due to the multi-label problem, values are specified on a class-wise. The values are normalized, and the color scale employs green to signify high values for True Positives and True Negatives, while lower values are represented in a reddish hue. Conversely, for False Negatives and False Positives, the color scale is applied in the opposite manner, indicating the performance and accuracy of classifications within the model visually.

**Evaluation.** In the evaluation stage of the ML pipeline, a code module is deployed to assess the prediction performance of the trained classifier using the test dataset set aside during the data splitting stage. To ensure a robust evaluation of performance, the entire ML pipeline was executed three times with different random seeds, and the results were aggregated.

As seen in Figure 6.17, the confusion matrix is subdivided class-wise for the multi-label case. This implies that for each class being observed, there is a $2 \times 2$ matrix in the figure, showing the normalized values of a binary classification of the model, indicating how well the model predicts the presence (positive, POS) or absence (negative, NG) of the respective label. The rows represent the values concerning the actual class (Ground Truth), and the column values represent the predicted labels.

For example, looking at the 'ADV_LD' matrix, the model correctly predicts the non-existence of the class in a sample (TN) 93% of the time. In 47% of cases, the model fails to predict a class when it exists (FN). The color scale is chosen so green indicates a high error-free result (TP and TN), while reddish coloring indicates a lower error-free value. The scale is the opposite for erroneous results (FP and FN). Thus, green $2 \times 2$ matrices suggest high model accuracy for this class, while matrices with yellow and red fields indicate classes with high misclassification by the model.

In addition to the confusion matrix, Table 6.7 lists the calculated values of Precision, Recall, and $F_1$-Score, as well as the number of available ECG samples in the test dataset for each class. The aggregated metrics presented in Table 6.8 include micro-, macro-, and weighted average values for metrics that can be calculated for multi-label classifiers.

The evaluation stage creates plots as artifacts for evaluating the model, as well as the calculated metrics and classification report, which can be loaded by the dashboard pod routines during deployment.

**Table 6.7:** Classification report of the ECG signal classifier. The table showcases the Precision, Recall, $F_1$-Score, and sample count for each class, with error margins of three runs, illustrating the balanced evaluation of the ECG signal classifier across the investigaded classses.

| | Precision | Recall | $F_1$-Score | # Samples |
|---|---|---|---|---|
| ADV_LD | 0.62 ± 0.03 | 0.52 ± 0.01 | 0.57 ± 0.02 | 1,891 |
| ADV_LV | 0.01 ± 0.01 | 0.01 ± 0.01 | 0.02 ± 0.01 | 86 |
| ADV_RD | 0.60 ± 0.02 | 0.55 ± 0.07 | 0.57 ± 0.03 | 82 |
| AFIB | 0.94 ± 0.00 | 0.88 ± 0.00 | 0.91 ± 0.00 | 730 |
| AFLUT | 0.71 ± 0.07 | 0.52 ± 0.16 | 0.59 ± 0.07 | 34 |
| AV1 | 0.63 ± 0.00 | 0.48 ± 0.01 | 0.54 ± 0.01 | 384 |
| ICD_NON | 0.38 ± 0.01 | 0.08 ± 0.02 | 0.13 ± 0.03 | 379 |
| IRBBB | 0.65 ± 0.00 | 0.63 ± 0.08 | 0.64 ± 0.04 | 538 |
| LAFB | 0.82 ± 0.01 | 0.75 ± 0.07 | 0.78 ± 0.04 | 782 |
| LBBB | 0.93 ± 0.01 | 0.86 ± 0.04 | 0.89 ± 0.02 | 259 |
| L_QT | 0.12 ± 0.10 | 0.17 ± 0.10 | 0.12 ± 0.10 | 53 |
| PM | 0.96 ± 0.00 | 0.78 ± 0.04 | 0.86 ± 0.02 | 144 |
| PVC | 0.82 ± 0.01 | 0.85 ± 0.00 | 0.83 ± 0.00 | 590 |
| RBBB | 0.81 ± 0.03 | 0.90 ± 0.01 | 0.85 ± 0.01 | 259 |
| SARRHY | 0.64 ± 0.01 | 0.53 ± 0.00 | 0.58 ± 0.00 | 370 |
| SNR | 0.94 ± 0.00 | 0.96 ± 0.00 | 0.95 ± 0.00 | 8,054 |
| SRB | 0.74 ± 0.02 | 0.50 ± 0.00 | 0.60 ± 0.01 | 307 |
| SRT | 0.87 ± 0.02 | 0.89 ± 0.01 | 0.88 ± 0.02 | 394 |
| SVES | 0.34 ± 0.04 | 0.26 ± 0.03 | 0.30 ± 0.03 | 192 |

**Table 6.8:** Aggregated model performance metrics of the ECG signal classifier. The table shows calculated the metrics Precision, Recall, $F_1$-Score, $MCC$ and $JSC$, each for three ML pipeline runs in micro, macro and weighted avarage.

| | micro | macro | weighted |
|---|---|---|---|
| Precision | 0.84 ± 0.00 | 0.65 ± 0.00 | 0.82 ± 0.00 |
| Recall | 0.80 ±0.01 | 0.58 ± 0.01 | 0.80 ± 0.01 |
| $F_1$-Score | 0.82 ± 0.01 | 0.61 ± 0.01 | 0.81 ± 0.01 |
| $MCC$ | 0.81 ± 0.01 | 0.59 ± 0.01 | 0.55 ± 0.02 |
| $JSC$ | 0.70 ± 0.01 | 0.49 ± 0.01 | 0.72 ± 0.01 |

Related to Gorodkin 2004, the Matthews Correlation Coefficient (MCC) for the multiclass case is definable using the confusion matrix $C$ with $K$ classes:

$$MCC = \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{(s^2 - \sum_k^K p_k^2) \times (s^2 - \sum_k^K t_k^2)}}$$

while $t_k$, $p_k$, $c$ and $s$ are intermediate variables defined as:

$t_k = \sum_i^K C_{ik}$

$p_k = \sum_i^K C_{ki}$

$c = \sum_k^K C_{kk}$

$s = \sum_i^K \sum_j^K C_{ij}$

The MCC yields a value within the interval $[-1, +1]$, whereby an MCC of +1 signifies an impeccable prediction congruent with the test dataset, whereas a value of 0 is indicative of a classification outcome akin to random chance.

As formulated by Jaccard 1901, the Jaccard Similarity Coefficient (JSC) can be applied for an multilabel classification case and measures similarity between a ground truth label set $y$ and the predicted label set $\hat{y}$:

$$JSC(y, \hat{y}) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|}$$

The JSC spans a continuum from 0 to 1, where 1 denotes a perfect match and 0 means no match between predicted and ground truth sets.

Based on the results of the evaluation stage, which detail the prediction performance of the ECG classification model on the entire dataset, they can again be used as a benchmark for AL simulations in Section 6.2.3.

**Validation.** In the validation stage, various model interpretation algorithms were implemented. The results are presented in detail in Chapter 8.

**Encoding.** To realize the density-based QS proposed by Beluch *et al.* 2018 in this application, the *Encoding* stage of the ML pipeline incorporates a dedicated logic for the computation of similarity vectors of samples. The method for generating sample representations, despite ECG signal data presenting different characteristics and structures from image data, can be applied to the used model architecture from Xu *et al.* 2020. By activating the last fully connected layer of the trained model, the output is transformed into vectors.

These vectors facilitate a comparison based on their Euclidean distance, adapting the technique described for determining essential similarity values of training samples in the implementation of the ML pipeline of the skin image analysis use case (see 6.1.2) (Sener *et al.* 2018). Storing calculated vectors as tracked data artifacts makes them immediately accessible during the execution of the query module, thus streamlining the querying process.

**Embedding.** To realize a cluster-based QS, an additional ML pipeline stage, *Embedding*, was introduced, implementing a separate code module for generating embedding vectors of the trained model using parametric Uniform Manifold Approximation and Projection (UMAP). This technique, proposed by Sainburg *et al.* 2021, transforms high-dimensional data into a lower dimension, aiming to preserve the key structures of the data.

The embedding vectors generated in this stage are utilized by the cluster-based QS, which is defined as a subsequent stage in the ML pipeline for the effective grouping of similar data points. Like the encoding vectors, storing the embedding vectors as tracked data artifacts ensures quick availability during the execution of the Query module.

**Query.** Given the complexity of the ECG signal classifier use case, which involved a larger dataset and a greater number of classes compared to the previous use case from Section 6.1.2, additional QSs were implemented to handle these challenges more effectively. The five selected QSs were chosen to provide a diverse set of approaches, and the hybrid QSs strike a balance between informativeness and representativeness in the query process.

- The implementation of a *Random* based sampling was adopted. It selects samples without following a specific pattern, thus providing a basis for comparison with other QSs in this application case.
- For information-based QS, the implementation of *Uncertainty* based sampling was further adopted. This strategy selects samples for which the model exhibits the highest uncertainty in its predictions, following the least confident approach by Lewis *et al.* 1994a (c.f. 2.1.3).
- As a representation-based QS, the approach proposed by Zhdanov 2019, utilizing *k-means* clustering, was implemented. Using the vectors stored during the *Embedding* stage, $k$ clusters are formed. Samples are then selected based on their maximum distance from the cluster center.
- The *Density Weighted Diversity (DWD)* based sampling proposed by Wang *et al.* 2021 was implemented as a hybrid QS, selecting samples based on both informativeness and representativeness. By weighing the diversity of samples against their density, this strategy achieves a balance between exploring the input space and exploiting dense areas.
- To balance the conflicting aspects of representation-based and information-based QSs, another hybrid QS, *QueryMerging*, was implemented. As shown in Algorithm 1, *QueryMerging* queries individual batches for each selected query function and then merges them according to the desired ratio. The final query set is proportionally mixed, taking into account the original ranking of individual samples in each query set.

This *Query* stage outputs a data artifact as a CSV file containing a configurable number of entries of sample_ids from the query set of

**Algorithm 1:** *QueryMerging.* The query set $\mathcal{B}$ of size $n$ would be assembled using the set of query functions $\mathbb{Q}$ in ratio $R$. $\mathcal{U}$ is the unlabeled data pool and $f$ is the model.

```
Input: $Q \leftarrow$ set of $k$ query functions
       $R \leftarrow$ set of query strategy ratios
       $n \leftarrow$ size of the query set
       $\mathcal{U} \leftarrow$ the unlabeled dataset
       $f \leftarrow$ the model
Output: $\mathcal{B} = \{s_1, s_2, ..., s_n\}$
ensure: $0 < n \leq |\mathcal{U}|$
initialize: $b_1, b_2, ..., b_k \leftarrow \emptyset$ and $B \leftarrow \{b_1, b_2, ..., b_k\}$
foreach query function $q_i \in \mathbb{Q}$ do
    initialize: ${q_i}^f$
    // acquire samples to temporary batches
    $b_i \leftarrow {q_i}^f(\mathcal{U}, n)$
initialize: $j \leftarrow 0, \mathcal{B} \leftarrow \emptyset$
// assemble the query set
do
    // Choose samples from each batch based on the given ratio
    $i \leftarrow \arg\max R$
    $m \leftarrow R_i$
    $\mathscr{b} \leftarrow \emptyset$
    foreach sample $s_l \in b_i$ do
        // add samples to query set only once
        if $s_l \notin \mathcal{B}$ then
            $\mathscr{b} \leftarrow \mathscr{b} \cup s_l$
            $j \leftarrow j + 1$
        // stop after sufficient sampling
        if $|\mathscr{b}| = m$ then
            break
    $\mathcal{B} \leftarrow \mathcal{B} \cup \mathscr{b}$
    $R_i \leftarrow 0$
while $j < n$;
```

samples selected by the applied QS. As described in Section 6.2.2, this artifact is loaded by the annotation widget of the annotation UI pod during deployment, making these samples available for the human oracle to annotate. For the simulation case, the query set file is read by the code module of LIFEDATAs experiment utils, whereby the sample's annotation is automatically generated by looking up the true label in each AL iteration.

### 6.2.3 Active Learning Simulation

Applying LIFEDATA in this use case, along with the availability of a fully annotated dataset, initially allowed for the investigation of the annotation process's efficiency without involving human feedback. To evaluate the AL process for an ECG signal classifier, the focus was on whether the application of AL in annotating ECG signal data could reduce the required annotation effort and determine which of the implemented QSs would be most effective in preferentially sampling rare classes over majority classes during the annotation process.

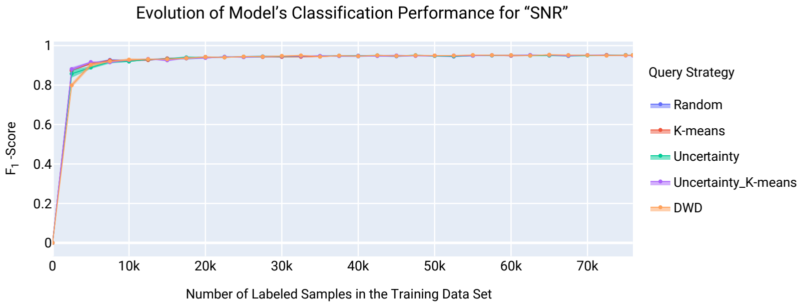


**Figure 6.18:** Classification performance of class 'SNR'.

The experimental approach simulated the labeling process to explore these aspects. It is important to note that the simulation was conducted under laboratory conditions, where the true label was determined by lookup, ensuring error-free annotation and excluding user errors.

The ML pipeline described in Section 6.2.2 formed the basis for implementing the simulation. Again, by following the methodology proposed in Section 4.3.3, sub-experiments in various experiment branches were initiated, each simulating the annotation process under the assumption that one of the implemented QSs with a configured query set size of 2,500 is active. The hybrid QS *QueryMerging* defined in Algorithm 1 was initialized with the $\mathbb{Q}$ functions of the information-based QS uncertainty based sampling, and the representation-based QS k-means in the balanced ratio of $R = [1 : 1]$. To validate the results, each sub-experiment, analogous to the approach in the ML pipeline implementation, was conducted with three different random seeds. The results of these runs were calculated for average and standard deviation.

The analysis of the predictive performance of the trained model throughout the annotation process is facilitated using the $F_1$-Score, which serves as a simplified performance metric. Figure 6.18 presents the average $F_1$-Score for the majority class 'SNR' in relation to the absolute number of annotated ECG signals. The colored lines represent the average value across the three sub-experiments for each QS, while the shaded areas indicate the standard deviation. Notably, the maximum predictive performance of the model for the 'SNR' class is achieved with a dataset of less than 30,000 labeled samples, corresponding to less than 50% of the data. This observation applies to all investigated QSs. None of the QSs resulted in a significantly different predictive performance of the model at the end of the simulations.

Figure 6.19 allows for a more detailed examination of performance per class. It shows the progression of $F_1$-Scores achieved by the model for the remaining 18 classes depending on the proportion of annotated data. Although the development of the $F_1$-Score in most classes correlates with the number of annotated data, it is noticeable that the 'ADV_LV' class is rarely correctly predicted at any point in time.

Moreover, the diagrams illustrate the different capabilities of the model in terms of its predictive ability for the respective classes. While some classes (such as 'AFIB', 'PVC', 'SRB') achieve high $F_1$-Scores early and thus appear easier to learn for the chosen model architecture in this setup, other classes (such as 'ICD_NON', 'LAFB') show low performance values or greater variances (e.g., 'AFLUT'). This observation may be attributable

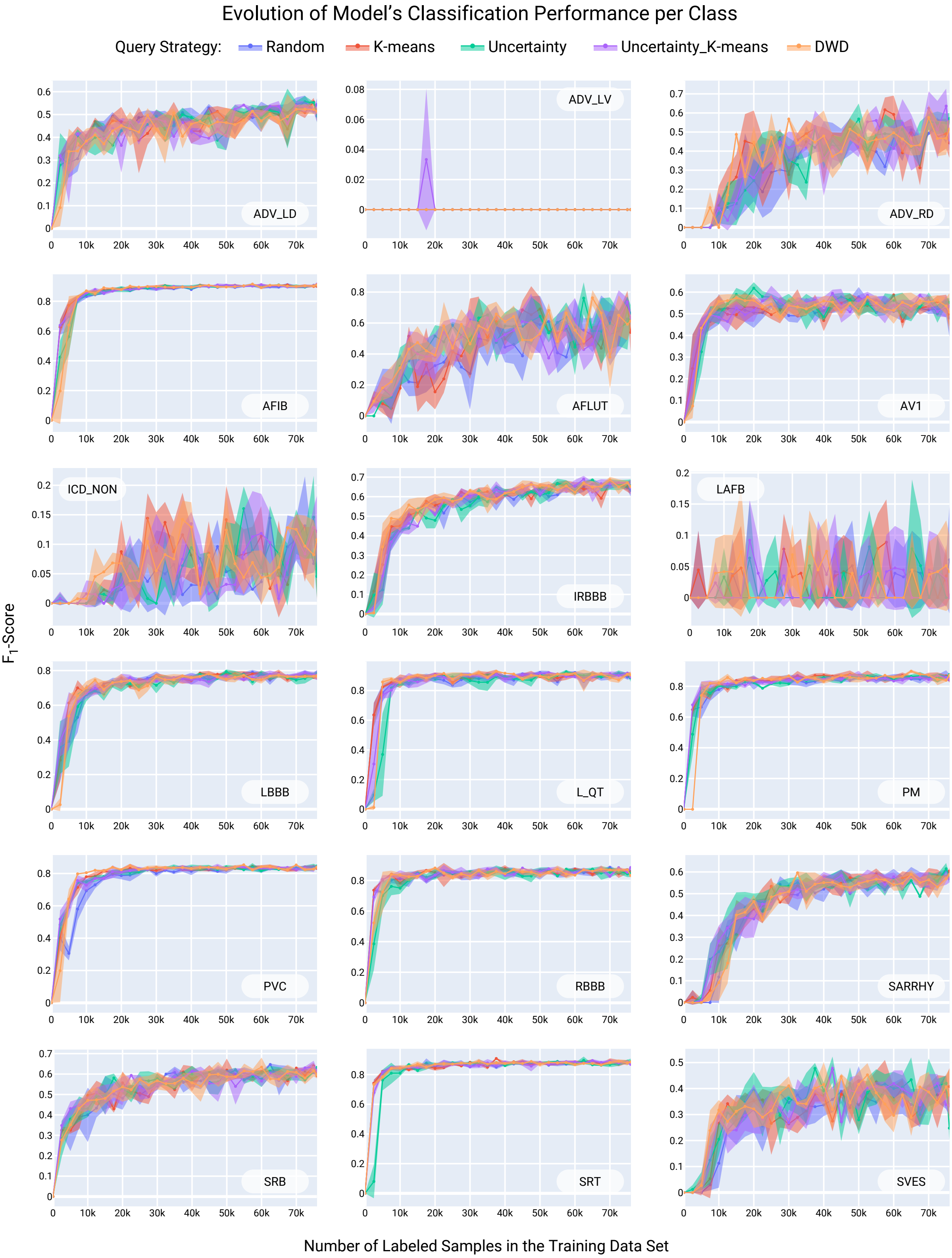


**Figure 6.19:** ECG signal classification performance per class.

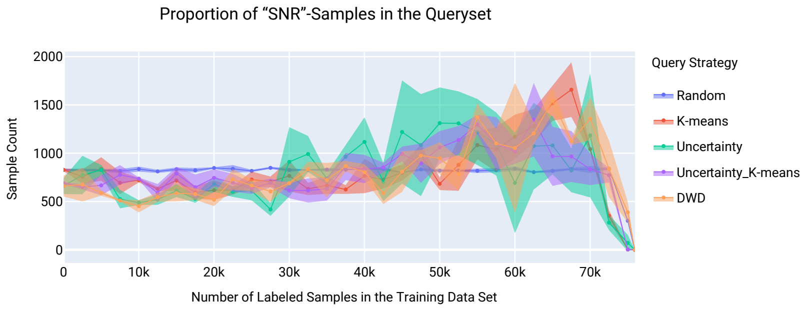


**Figure 6.20:** Query set proportion of class 'SNR'.

to a higher complexity of the inherent patterns of these classes or to the low number of existing samples.

A comparison of the QSs for this use case reveals that none of the implemented approaches result in a significant efficiency increase in the model's achievable predictive performance. The model appears to exhibit similar learning behavior across all QSs throughout the simulated annotation process, indicating that no QS led to a notable difference in the model's predictive performance.

To evaluate the AL iterations concerning the implemented QSs and class selection, analyses of the frequency distribution of the created query sets are conducted. This is facilitated again by the established project setup and the integration of the data version control system into LIFEDATA, allowing the query sets created in the simulated AL iterations to be downloaded and evaluated as artifacts from the data remote.

Figure 6.20 initially provides an overview of the proportion of the queried class SNR within the various QSs. In this particular use case, the intuition is to minimize the selection of this class to the greatest extent feasible. Considering annotation costs and the high performance of the classifier for this class, the goal is to present the human annotator with other classes in the data labeling process as much as possible.

In Figure 6.20, the sample count (ordinate) is set in relation to the number of labeled samples in the training dataset (abscissa). The blue line, representing random sampling, serves as the baseline, corresponding to random selection. It is observed that the four other implemented QSs offer a significant advantage over random sampling. Only after about halfway through the annotation process does the proportion of samples with the class 'SNR' in the query set increase, indicating that the samples of the other classes were annotated by this point, which is why the frequency distribution curve for the class 'SNR' rises.

The remaining 18 classes are compared in Figure 6.21. The individual plots are similarly structured and show the proportion of each specified class in the query set throughout the simulated annotation. Here, the curves of the rarer classes show the expected opposite trend. In the first half of the annotation process, the proportion of samples of almost every class is higher than random-based sampling. It steadily decreases with an increasing proportion of the total labeled training data volume. An exception is the classes 'SRB' and 'SRT', which show a similar trend curve to 'SNR', possibly explained by the generally higher total proportion of these classes in the dataset, which is evident from the scaling of the x-axis of these subplots.

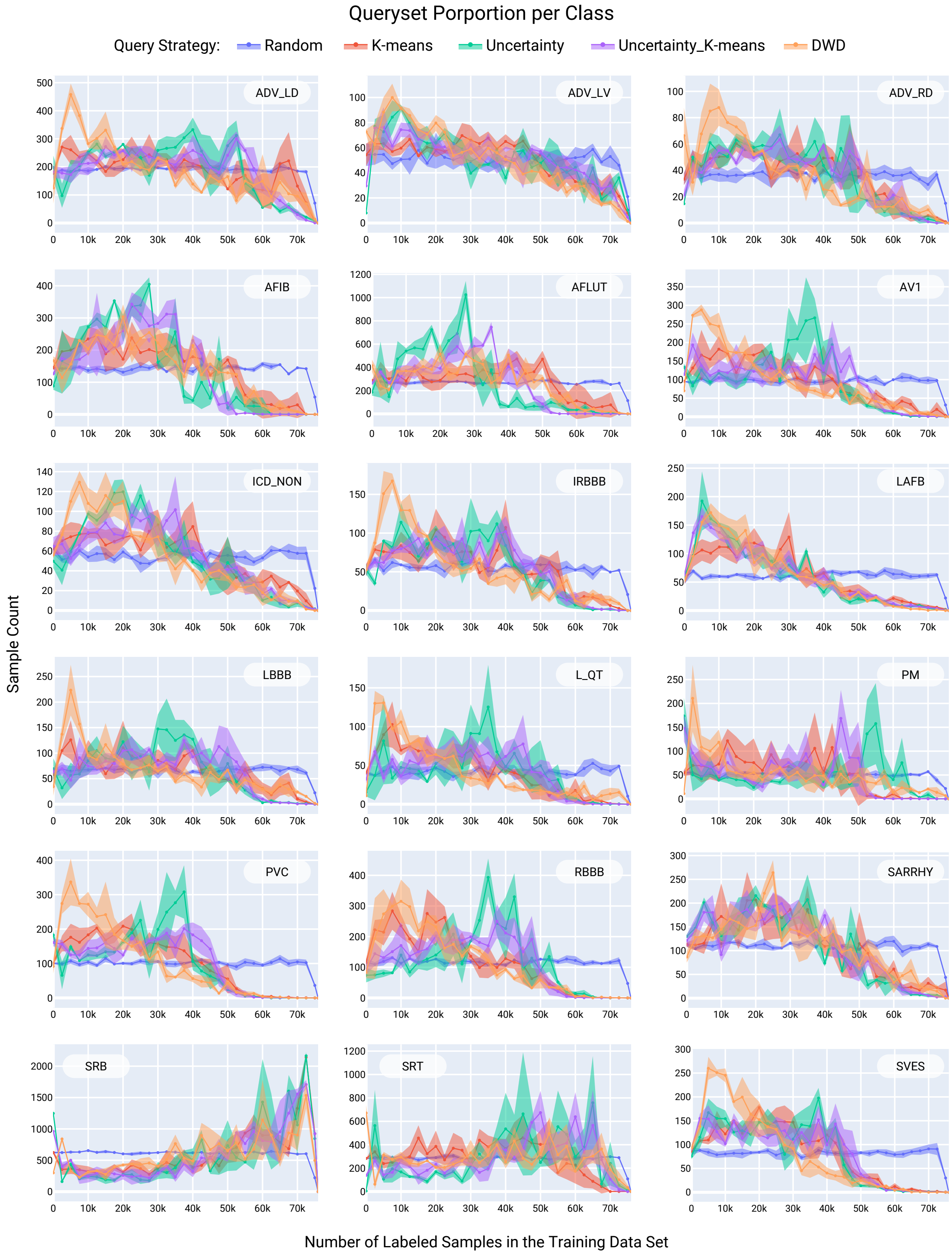


**Figure 6.21:** Query set proportion per class.

Comparing the different QSs, the DWD-based sampling, for example, for the classes 'ADV_LD', 'ADV_RD,' and 'PVC,' shows a significant advantage, as the proportion of these classes in the query set of the first iterations is considerably higher compared to the other QSs. For the class 'AFLUT', the frequency distribution of the query set suggests that uncertainty-based sampling offers a clear advantage.

In summary, the implemented QSs are equally suitable for ECG signals and offer a significant advantage in selecting rare classes over random-based sampling. In the simulation, only the classes 'SRB' and 'SRT' deviate from this observation, likely due to their high proportion in the dataset used. Since the implemented QSs perform differently, the choice of which to use depends on which classes should be prioritized. The DWD QS provides a balanced advantage by achieving a preferred selection for several rare classes early in the annotation process.

### 6.2.4 Human-in-the-loop Study

While the simulation-based analysis has demonstrated the effectiveness of various querying strategies, highlighting the efficiency in the annotation process, the focus in the subsequent section shifts to an equally critical element of the AL learning cycle: human-provided annotation. Here, we focus on the integrity and consistency of the labels provided by human annotators in this AL setup.

**Study Design and Implementation.** Figure 6.22 illustrates the methodological approach of the designed study. Initially, a cardiology expert identified representative ECG signals using the QS DWD. The class selection was increased from the original 19 to a total of 40 classes, which were categorized by the domain expert into superclasses („general diagnosis“) and subclasses („specialized diagnosis“). These manually selected samples formed the static test set, which was systematically integrated into the annotators' workflow over four months.

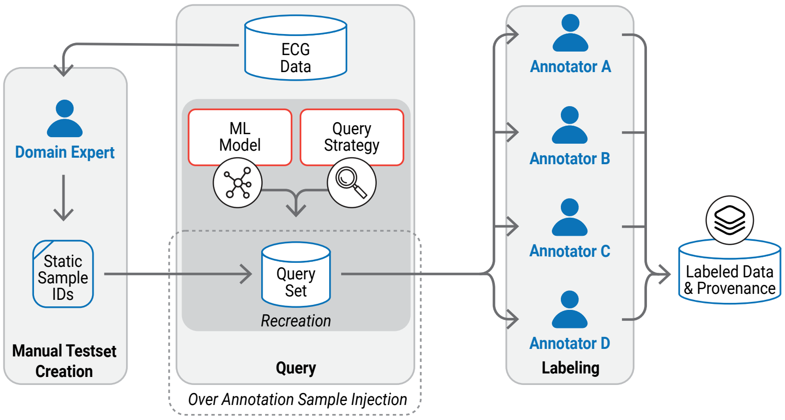


**Figure 6.22:** Implementation of the inter- and intra-annotator agreement study. At the beginning of the study, a domain expert created a manual test set. The samples from this test set were added to the query set by four annotators over a period of four months. The collected labels were then available for evaluation.

The four annotators involved, all similarly qualified with a degree in human medicine, labeled these samples along with others from the dynamically generated query set. The annotators were not informed that these samples had been manually selected for annotation, so they assumed they were annotating samples selected by the model's QS, as usual. Each of the four annotators was presented with 40 samples from the static test set for re-annotation in each of the four consecutive months. In addition, the annotators had access to an annotation guide in the GUI provided by LIFEDATA, which included detailed guidelines for assigning possible classes.

To replicate the process shown in Figure 6.22 in the functionality of LIFEDATA, the API to the core framework was extended with a method for querying the static test set, whose function was implemented in the data persistence layer, specifically in the `DBAssignmentRepository` (see Listing 5.5). This extension enabled the targeted injection of the static sample_IDs, which are managed in an index file within the project instance, thus ensuring that each annotator was presented with these ECG signals for labeling in the GUI during their monthly annotation activity.

**Analysis.** The information on labels and provenance that accumulated during the study period was collected in LIFEDATA's relational DB and subsequently analyzed using the DB export function provided by LIFEDATA. To assess label quality, the metric introduced by Krippendorff 1970, Krippendorff's Alpha, was used - a reliability coefficient that enables quantifying the degree of agreement among given labels, thereby determining the reliability of annotations generated in the AL environment.

Krippendorff's Alpha facilitated the determination of both inter-annotator agreement and intra-annotator agreement. Inter-annotator agreement measures the consensus among annotators and was assessed by comparing the labels assigned to the same samples by different annotators. In other words, inter-annotator agreement describes the degree to which, for example, *Annotator A* and *Annotator B* agree on a label for a sample.

Intra-annotator agreement, referred to as 'self-agreement', evaluates the consistency of labels that a sample is given by the same annotator across different annotation iterations, i.e., the deviation, for example, of the labels given to a sample by *Annotator A* to the same labels given to the same sample by *Annotator A* upon re-labeling.

Krippendorff's Alpha ($\alpha$) (Krippendorff 1970) is defined as:

$$\alpha = 1 - \frac{D_o}{D_e}$$

where $D_o$ is the observed disagreement,

$$D_o = \frac{1}{N} \sum_{c,c'} N(c, c') \cdot \delta^2(c, c')$$

and $D_e$ is the expected disagreement due to chance:

$$D_e = \frac{1}{N(N-1)} \sum_{c,c'} N(c) \cdot N(c') \cdot \delta^2(c, c')$$

Here, $N$ is the frequency of samples, $N(c, c')$ is the number of values in the resulting coincidence matrix for the classes $c$ and $c'$. $\delta(c, c')$ is a difference function. For the multi-label case, we can use the $JSC$ as defined in 6.2.2, which gives us the function with:

$$\delta(c, c') = 1 - \frac{|c \cap c'|}{|c \cup c'|}$$

**Results.** The systematic evaluation of the submitted annotations over a period of four months yielded the agreement values listed in Table 6.9. A consistent pattern in agreement was observed in both the subclass and superclass level assessments. The Alpha inter-values for subclasses were in the range of $\alpha = 0.33$ to $\alpha = 0.34$, while the values for superclasses were slightly higher, ranging between $\alpha = 0.35$ and $\alpha = 0.37$. Although the values showed consistency over the months, suggesting that the annotators' agreement remained stable over time, the moderate values indicate the difficulty of achieving high agreement in this annotation task, which may be reflected by the minimal differences between superclass and subclass - hinting at a high degree of complexity, precision, and inherent interpretive space of ECG signal data.

In examining the agreement between annotators, as depicted in Table 6.10, there was measurable variance among the annotators, with values ranging from $\alpha = 0.56$ to $\alpha = 0.63$ for subclasses and between $\alpha = 0.57$ and

| Month | Annotated samples | Alpha inter (subclass) | Alpha inter (superclass) |
|---|---|---|---|
| 1 | 160 | 0,34 | 0,36 |
| 2 | 160 | 0,33 | 0,36 |
| 3 | 160 | 0,34 | 0,37 |
| 4 | 160 | 0,33 | 0,35 |

**Table 6.9:** Inter-annotator-agreement of the four annotators.

$\alpha = 0.63$ for superclasses. *Annotator A* and *Annotator B* showed higher self-agreement with Alpha intra-values $\geq \alpha = 0.60$. Conversely, *Annotator C* and *Annotator D* demonstrated slightly lower agreement. These deviations could point to differences in individual annotators' labeling consistency or their interpretation of the annotation guidelines.

| Annotator | Alpha intra (subclass) | Alpha intra (superclass) |
|---|---|---|
| Annotator A | 0,62 | 0,62 |
| Annotator B | 0,61 | 0,63 |
| Annotator C | 0,56 | 0,57 |
| Annotator D | 0,56 | 0,57 |

**Table 6.10:** Self-agreement of the four annotators.

Krippendorff's Alpha can be interpreted in such a way that $\alpha = 1$ indicates perfect agreement. In contrast, $\alpha = 0$ denotes no agreement comparable to what could be achieved by chance. Conversely, a negative value $\alpha < 0$ suggests systematic disagreement, implying that the annotators consistently annotate differently. (Krippendorff 2011)

However, as mentioned by Passonneau 2006, the evaluation of Krippendorff's Alpha leaves open what value of reliability is good enough and what decisions can be derived from it. Since there is no universally applicable threshold, it is, therefore, difficult to determine which value is acceptable for conclusions.(Passonneau 2006)

In summary, the assessment of Inter- and Intra-Annotator Agreement using Krippendorff's Alpha metric shows a higher agreement among annotators compared to the agreement between different annotators. Annotators tended to be more consistent with themselves than with other Annotators, which could be attributed to the annotation task requiring a subjective judgment despite the labeling guidelines. Overall, the results of the calculated Krippendorff's Alpha values suggest a moderate reliability of the annotations.

## 6.3 Summary

This chapter described the application of the LIFEDATA framework to realize AL projects in two life sciece use cases.

In Section 6.1 a scenario was demonstrated in which an ML pipeline is used to train a skin image classifier. Dermatoscopic images from the HAM10000 dataset (Tschandl *et al.* 2018) were leveraged, with the initial ML pipeline proposed in the LIFEDATA project template being tailored to fit this specific application. This involved incorporating a logic for image data preprocessing and implementing a DNN based on the DenseNet201 architecture (Huang *et al.* 2017).

The resultant classification model was evaluated through multiple iterations of experimental ML pipeline runs. Three distinct QSs were implemented to probe the facets of the AL loop. Simulations highlighted that employing uncertainty or density-based sampling, in contrast to random selection in the annotation process, led to more strategic queries, thereby effectively harnessing human resources in case of a highly imbalanced dataset.

LIFEDATA's architecture was instrumental in fostering a robust and scalable project setup. Consistent code and data artifact versioning ensured all experiments' reproducibility. Furthermore, it enables the assessment of compute-intensive AL simulations by making the requisite data artifacts accessible in data repositories and within the experiment tracking service.

This was emphasized in the application of LIFEDATA to the second use case in Section 6.2, in which a scenario was implemented with an ML pipeline aiming to classify multi-label arrhythmia in ECG signal data. Multiple suitable 12-lead ECG datasets were carefully selected from the PhysioNet Challenges 2020 (Alday *et al.* 2020) and 2021 (Reyna *et al.* 2021) for this use case. Subsequently, a DNN model employing an architecture proposed by Xu *et al.* 2020 was trained.

The architecture of the ML pipeline was expanded to incorporate the necessary logic for preprocessing the raw signals, along with additional steps that enable the efficient implementation of (besides information-based) representation-based QSs. Furthermore, the query module was enhanced by introducing a hybrid QS, which was evaluated in AL simulations regarding the efficiency of the annotation process.

The provision of a GUI for annotating samples and the realization of a customized dashboard GUI, which visualizes metrics of the annotation process, facilitated the implementation of a labeling scenario with real humans in the AL cycle. A study investigated the reliability of the annotations by calculating the degree of inter- and intra-annotator agreement.

The increased complexity of this project required an adaptation of the infrastructural setup, where the components provided by the LIFEDATA project instance could be extended. This demonstrated the adaptability of the framework and its ability to scale with increasing requirements.

# IV. Interpretable Machine Learning An Integration Approach

# Domain Specific Model Interpretations 7

In the pursuit of implementing trustworthy AI systems, the aspect of explainability regarding the functioning of deployed models leads to the concept of XAI, which aims to translate a system's outputs or decisions into a form understandable by humans. In this context, domain-specific approaches in XAI play an increasingly important role as they enable the development of explanation methods tailored to the specific characteristics of particular application fields (Murdoch *et al.* 2019).

An example where the need for XAI becomes apparent was shown in Section 6.1, situated in the application field of AI-based CDSS in medical diagnostics for the classification of skin lesions. Such CDSS often utilize complex ML models, which pose challenges in interpreting their decisions. The complexity and opacity of these models can make it difficult for users to understand and verify the diagnostic suggestions provided by the CDSS (Lucieri *et al.* 2022).

Here, just like the model itself, explanations must be adapted to the problem to be useful for the specific application case (Miller 2019). The integration of domain-specific knowledge, as illustrated by dermatology, enables dermatologists to explain the decisions of AI systems in a manner understandable from their relevant perspective.

The following chapter introduces a domain-specific concept for this purpose, first presented in the publication Stieler *et al.* 2021. This approach synthesizes the interpretable ML method LIME, proposed by Ribeiro *et al.* 2016, with the human-applied ABCD rule, a diagnostic procedure for differentiating between melanocytic and non-melanocytic skin lesions (Stolz *et al.* 1991).

As introduced in Section 7.3, the perturbation algorithm of LIME is modified along two of the dimensions of the ABCD rule. To validate the degree of significance of the explanations, medically irrelevant perturbations are introduced in addition to the medically relevant dimensions. Section 7.4 demonstrates the application to a black-box model and formulates hypotheses about its predictions. The observations are presented in Section 7.5 and subsequently discussed in Section 7.6. First, Sections 7.1 and 7.2 briefly overview related works and their methodology.

**7.1 Interpretable Machine Learning in Skin Image Analysis . . . . . . . . . . 186**
**7.2 Dermatologist's Human Approach . . . . . . . . . 188**
**7.3 Explainer for Skin Image Classifier . . . . . . . . . 189**
7.3.1 Perturbation Dimensions 190
7.3.2 Implementation . . . . . 191
**7.4 Application . . . . . . . . 193**
7.4.1 ML Pipeline Adaption and Experimental Setup 193
7.4.2 Hypothesis . . . . . . . . 194
**7.5 Empirical Results . . . . 195**
7.5.1 True Positives . . . . . . 195
7.5.2 False Negatives . . . . . . 197
7.5.3 Aggregation . . . . . . . 198
**7.6 Discussion . . . . . . . . 200**
**7.7 Summary . . . . . . . . . 201**

## 7.1 Interpretable Machine Learning in Skin Image Analysis

In the skin image analysis domain, the interpretability of ML models is a crucial factor for fostering acceptance and trust among dermatologists and patients in AI-supported diagnostic procedures. The uniqueness often lies in the complex decisions of algorithms applied in a field where visual pattern recognition and subtle nuances of morphology are decisive. XAI methods allow for subtly examining features an ML model considers in its predictions.

An example of implementing an XAI method in AI-supported skin image analysis is provided by Winkler *et al.* 2019, who visualized the outputs of a trained CNN using saliency maps (Simonyan *et al.* 2014). They found that annotations made by doctors on dermatoscopy images could influence the ML model's predictions. These artifacts introduced by doctor interaction led to the ML model learning features relevant to the Melanoma class, thereby not learning the medically relevant concept.

Although most XAI methods are possible without domain-specific adjustments and can increase the interpretability of model predictions, various works show that integrating medical expertise into implementing XAI can increase acceptance and understanding. Wang *et al.* 2022 highlighted the added value of incorporating medical knowledge into the conception of XAI models. They show that the effectiveness of XAI applications depends on the technical implementation as well as on integrating this knowledge into model explanations.

The application of domain-specific features in developing a transparent ML model for melanoma diagnosis is further investigated by Gareau *et al.* 2020. Their approach is based on the use of Imaging Biomarker Clues (IBCs) to provide better interpretability for doctors. In their study, an ensemble model was trained using 38 IBCs that quantify visually relevant features, representing intuitive visual clues that enable doctors to validate the model's diagnostic decisions based on visible and understandable characteristics.

Lucieri *et al.* 2020 investigated the interpretability of a DL-based skin lesion classifier using Concept Activation Vectors (CAVs, Kim *et al.* 2018). They examined whether an ML model learns and uses disease-related concepts similar to those described and used by dermatologists, with results suggesting that the model indeed utilizes disease-related concepts in prediction.

The study by Khater *et al.* 2023 focuses on implementing XAI techniques using SHAP (Lundberg *et al.* 2017) in decision tree-based models trained with extracted features. SHAP was used to post-hoc indicate the importance of the features used in the model's predictions, such as pigment network and count of different colors in the lesion.

The application of Class Activation Mapping (CAM, Zhou *et al.* 2016) to highlight diagnostically relevant areas in histological images was implemented by Jiang *et al.* 2021. This approach, shown by Mridha *et al.* 2023 in the form of Gradient-weighted Class Activation Mapping (GradCAM, Selvaraju *et al.* 2017) on dermatological images, aims to enhance the interpretability of the CNN models trained by the authors.

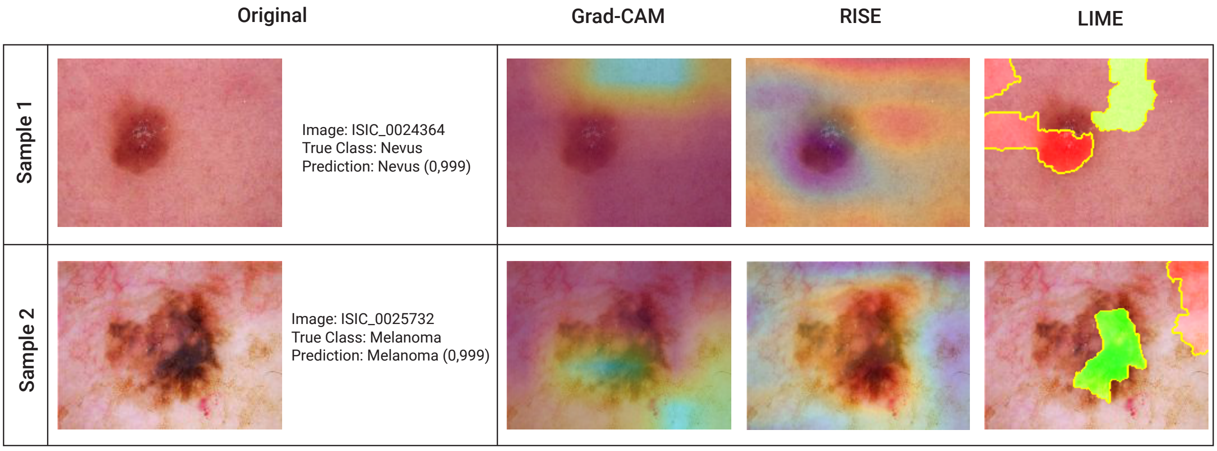


**Figure 7.1:** Local model output interpretations of the DNN-based classifier. Correct model predictions of two test samples are analyzed by three different model interpretation methods. Colored overlays indicate the degree of importance in relation to the predicted class.

This interpretation mechanism is limited to CNN models, using the gradient flowing into the last convolutional layer to highlight regions in the image that are important for prediction.

Randomized Input Sampling for Explanations (RISE) is a model-agnostic approach for generating local explanations for image data based on the principle of occlusion. Random masks are generated to cover image areas, pixels, for a specific pattern. To create an explanation, the pattern is occluded with these masks, and model predictions are determined. Results are combined by calculating the importance of each pixel of the input image in relation to the resulting classification.

Xiang *et al.* 2020 implemented the LIME approach introduced by Ribeiro *et al.* 2016 in their ML pipeline for classifying skin lesions. As highlighted in this chapter, this surrogate model-based method generates a dataset of perturbed instances for the sample to be explained. Predictions of the perturbed data using the black-box model are weighted, and an interpretable local model is trained.

For image data, this involves occluding parts of a specific pattern, typically selecting such parts with superpixels, by default using the Quick-Shift algorithm (Vedaldi *et al.* 2008), which can lead to the generation of potentially unusable areas in the medical context (Schallner *et al.* 2020; Magesh *et al.* 2020). This challenge was further found by Xiang *et al.* 2020, noting that such a model interpretation method can show meaningful areas in a given sample but may lack specificity for both machines and humans.

Figure 7.1 applied three methods to a DNN model, GradCAM (Selvaraju *et al.* 2017), RISE (Petsiuk *et al.* 2018), and LIME (Ribeiro *et al.* 2016), each in their standard implementations without domain-specific adjustments. Although the model's classification for the two test patterns shown is likely correct, the explanation outputs show a fatal correlation: For **Sample 2**, relevant areas of a melanoma are marked. **Sample 1** shows a nevus, and the important areas for the model lie outside the lesion. A critical feature seems to be the skin, not contributing to the prediction.

The simple application of such model interpretation methods in an AI system already demonstrates their potential. Rather than relying solely on prediction, the results provide insight into how the underlying model arrived at its decision. However, all these outputs from various methods only show image areas, the significance of which can vary greatly. There is a clear lack of domain-specific contexts: interpreting the results of these methods requires significant training for professionals. Hauser *et al.* 2022 conclude in their systematic review on XAI in skin cancer recognition the importance of collaboration with dermatologists for developing specific XAI methods to be used in AI-supported CDSSs.

## 7.2 Dermatologist's Human Approach

There are a variety of methods for diagnosing melanoma through human pattern recognition, with visual inspection often being the initial step in evaluating skin lesions. Over time, dermatologists and non-specialized physicians have established and applied various approaches.

*Pattern Analysis* focuses on identifying recurring patterns in skin lesions, requiring deep experience and understanding of the patterns in benign and malignant lesions. This technique introduced by Pehamberger *et al.* 1987 allows for a differentiated assessment by identifying specific pattern structures, though its complexity demands the interpretation of a wide array of patterns.

The *CASH* method, an acronym for Color, Architecture, Symmetry, and Homogeneity, provides a structured approach to visually evaluating skin lesions. Proposed by Henning *et al.* 2007, it analyzes the lesion's overall appearance based on specific characteristics. While enabling comprehensive evaluation, *CASH* necessitates thorough training for practical application, with outcomes largely dependent on the user's experience.

Some factors used in the *CASH* method trace back to the *7-Point Checklist*, developed by MacKie 1984. This method encompasses seven specific criteria to identify a melanoma, including three major features (Atypical Pigment Network, Blue Whitish Veil, Atypical Vascular Patterns) and four minor ones (Irregular Streaks, Irregular Dots, and Spots, Regression Structures, Irregular Diffuse Pigmentation), with a higher total score indicating a higher melanoma risk.

Friedman *et al.* 1985 proposed a method that further developed the idea behind the *CASH* method into a framework aimed at self-assessment by laypeople. This framework was later evolved into the widely known *ABCD rule* of dermatoscopy by Stolz *et al.* 1991, which was evaluated multiple times, including by Nachbar *et al.* 1994 and Rigel *et al.* 2010, reporting a sensitivity of ≈ 84% and specificity of ≈ 83.5%.

While the works of Annessi *et al.* 2007 and Rigel *et al.* 2010 provide a comprehensive comparison of the classification performance achievable by various human-based approaches, Gareau *et al.* 2020 compare these methods against several machine-augmented pattern recognitions, showing the *ABCD rule* outperforms most other human- as well as ML-based approaches on the ROC curves.

As outlined by Nachbar *et al.* 1994 the medical algorithm for visually distinguishing between melanocytic and non-melanocytic lesions relies

Sensitivity ($Se$) quantifies the ability of a test to correctly identify patients with a specific disease. It is calculated in the same way as the precision given in 6.1.2:

$$Se = P = \frac{TP}{TP + TN}$$

whereby in the medical context a high sensitivity is of decisive importance for the early detection of diseases and the minimization of misdiagnoses. Specificity ($Sp$) evaluates the accuracy of a test in confirming the absence of a disease. Defined as

$$Sp = \frac{TN}{TN + FP}$$

it indicates the accuracy of the test in avoiding FP results. High specificity ensures that patients without the disease are correctly identified.

on a multivariate analysis of four criteria, each scored semi-quantitatively to assess the likelihood of melanoma:

- Ⓐ Asymmetry: A lesion is examined along two perpendicular axes to determine its symmetry. A score of 0 indicates symmetry along both axes, 1 for asymmetry in one axis and 2 for asymmetry in both axes.
- Ⓑ Border: The abrupt cutoff of the pigment pattern at the lesion's border is assessed. The lesion is divided into eight segments, with a point added for each segment displaying an abrupt cutoff.
- Ⓒ Color: The presence and variety of colors within the lesion are evaluated, with potential colors including white, red, light and dark brown, blue-gray, and black. The color score ranges from 1 (single color) to 6 (six different colors).
- Ⓓ Differential Structure: This criterion considers the variety of structural components visible in the lesion, such as a pigment network, homogeneous areas, dots, globules, and streaks. The score can range from 1 (one structure type) to 5 (five different structure types).(Nachbar *et al.* 1994)

In simplified terms, the lesion is examined for all four criteria separately. The higher the score of a criterion applies to the lesion, the more likely it is to be classified as melanocytic. The sum of the scores finally leads to a diagnosis.

From this we conclude the *ABCD rule* is particularly suited for use as a human-friendly explainability method for two reasons: First, this approach not only leads to accurate classifications, it is easy to understand for humans, which means that it can be applied by physicians and to a certain extent by patients themselves. Second, the characteristics used to classify the lesion can be scored independently. Conversely, this has the effect that the four ABCD dimensions can be studied independently. In theory, adding or removing features in the dimensions directly impacts the classification.

## 7.3 Explainer for Skin Image Classifier

The following section introduces the integration of a model interpretation method with the previously presented established medical algorithm of the ABCD rule, defined by Nachbar *et al.* 1994, and the approach suggested by Ribeiro *et al.* 2016, LIME. The advantage of LIME lies in its perturbation-based strategy, offering interpretability for image data across various model architectures.

The essence of LIME's explanatory power resides in minimizing the complexity function $\Omega$ of the interpretable model $g$, where $\Omega(g)$ is inversely proportional to the model's interpretability to humans, which is significantly influenced by the number of features, $K$.

To establish the practical relevance of this method, the domain-specific explanator combines the theoretical foundations of LIME with the pragmatic approach of the ABCD rule for assessing skin lesions. The conventional LIME approach for image data is based on creating a perturbation dataset by pixel-wise modification, mainly by changing superpixels. Using the regularization path (Efron *et al.* 2004), $K$ image regions are selected, and a binary vector is generated, where 1 indicates the original superpixel, and 0 indicates a grayed-out superpixel - a procedure that Ribeiro *et al.* 2016 calls *K-Lasso*.

Although this method is generally applicable, it fails to capture the area-specific nuances essential for medical diagnosis. The modified methodology adjusts the images of skin lesions according to $K$ diagnostic criteria that follow the ABCD rule, an adaptation that preserves the interpretative strengths of LIME and further embeds it in the specific assessment framework of dermatologists.

### 7.3.1 Perturbation Dimensions

In realizing the perturbation logic, a focused assessment of each characteristic ensures that perturbations are restricted to one single dimension at a time in the input image, preventing alterations in features of another dimension. This is underpinned by the straightforward quantification and high diagnostic value, leading to an initial concentration on two ABCD dimensions: Ⓑ Boundary and Ⓒ Color, as they are the most prominent and objectively measurable markers.

Following research from Fong *et al.* 2017, two additional dimensions, Ⓡ Rotation and Ⓢ Shift, are incorporated to investigate the explanatory importance, introducing perturbations in medically irrelevant ways without affecting medically relevant features.

Figure 7.2 illustrates four dimensions with perturbed images in their strongest manifestations of each dimension. The image manipulations were implemented using Scikit's library for image processing (Van der Walt *et al.* 2014), which are artifacts and may look artificial to a human. However, we have to recognize that the particular characteristic is to be exaggerated. In the following, we go into detail how the perturbation is generated.

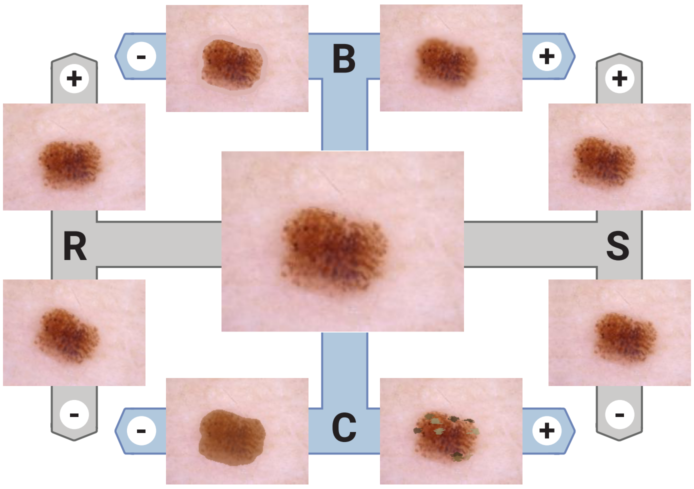


**Figure 7.2:** Perturbation dimensions of the explainer. The original image in the center is perturbed along medically relevant (blue) dimensions Ⓑ Boundary and Ⓒ Color, as well as medically irrelevant (gray) dimensions Ⓡ Rotate and Ⓢ Shift, each in a reinforcing (**positive**) and weakening (**negative**) manner.

**Ⓑ - Boundary** The implementation of the medically relevant dimension is realized along the **negative Boundary** direction by extracting the border area of the segmentation and drawing a sharply delineated line around the lesion. The color of this line corresponds to the average color values of the surrounding image areas and it is ensured that no artifacts arise in relation to the color which is used. To influence in the **positive Boundary** direction, the edge region is extracted from the segment and a Gaussian blur is added. This causes pixel values to fade into each other and the transition between lesion and skin is less sharply delimited.

**Ⓒ - Color** In the perturbed images of the **negative Color** dimension, the area within the segmentation of the lesion is turned into a uniform color. Possible color irregularities are thus harmonized. The coloring is transparent such that possible structures in the lesion are kept intact. Adding random color patches in the lesion area produces variation for the **positive Color** direction. They vary in size and color while ensuring that the color patches are transparent and possible structures remain recognizable, similar to the procedure towards the negative direction. The chosen colors correspond to plausible shades of brown, and their RGB color values were defined by a dermatologist for this implementation.

**Ⓡ - Rotate** This perturbation dimension is realized by rotating the sample by a given range of degrees. The range of values corresponds to the **positive** (left) and **negative** (right) direction. We chose mode 'reflect' as padding strategy, which mirrors neighboring pixel values along the vector.

**Ⓢ - Shift** An affine transformation is performed to shift the skin image. The translation parameter indicates the direction, which is increased with strengthening in the **positive** (left) or **negative** (right) direction. Same as for rotation, 'reflect' is used to pad the resulting gaps.

### 7.3.2 Implementation

The agnostic design of LIME enables the integration of the defined pertubation dimensions Ⓑ, Ⓒ, Ⓡ, Ⓢ - both in positive and negative direction. Before we go into the technical implementation, we adapt LIME's algorithmic logic, as originally defined by Ribeiro *et al.* 2016.

**Guided Perturbation Sampling.**

As described by Ribeiro *et al.* 2016, an explanation generated by LIME can be represented by the objective function that minimizes the measure $\mathcal{L}(f, g, \pi_x)$ (cf. 2.26). This measure indicates the infidelity of the linear model $g$ in approximating the complex model $f$ within the locality defined by $\pi_x$. Here, $\pi_x(x')$ assesses the proximity between a perturbed instance $x'$ and the original instance $x$. The approximation of $\mathcal{L}(f, g, \pi_x)$ in Ribeiro's proposed algorithm is achieved by sampling instances weighted according to $\pi_x$, where the weighting reflects the importance of proximity to the original instance $x$.(Ribeiro *et al.* 2016)

Although random samples are drawn from the input space, the weighting of these samples is not uniform but is based on their proximity to $x$, allowing closer samples to contribute more to the model approximation. The original implementation of LIME assumes random sampling, with the selective weighting of the samples playing a crucial role in ensuring the local fidelity of the simplified model.(Ribeiro *et al.* 2016)

We capitalize on this property by generating a set of $n$ perturbed instances $\{X'_B, X'_C, X'_R, X'_S\}$ for each lesion image, represented by $x$, so that each $x'_D$ is a variant of $x$ where one of the Ⓑ, Ⓒ, Ⓡ, Ⓢ attributes is either strengthened or weakened. For a given lesion image $x$ and a dimension $D = \{B, C, R, S\}$, the perturbation function $P$ creates a perturbed instance $x'_D$ by enhancing or reducing the feature associated with $D$, expressed as:

$$x'_D = P(x, D, \delta) \tag{7.1}$$

where $\delta$ corresponds to the originally intended weighting measure $\pi$ and assumes a value in range of $[-1, +1]$, indicative of the enhancement or reduction of the feature $D$, and the interval limits are represented as the strongest expression of the respective dimension.

**Generation of Sparse Linear Explanations.**

Following the creation of the perturbed samples $X'D$, the continuation of the implementation is based on the methodology of sparse linear explanations shown by Ribeiro *et al.* 2016. This step is instrumental in realizing LIME's strategy to identify an interpretable model via an interpretable representation that maintains local fidelity to the complex black box model. Owing to its sparsity, such a model provides interpretive insights into the features that significantly influence the predictions of the complex model for a specific instance.

The sparse linear explanation for a particular dimension $D$ of the original instance $x$ is captured by the function $\xi_D(x)$, which is derived from the optimization problem defined by Ribeiro and adapted as follows:

$$\xi_D(x) = \arg\min_{g \in G} \sum_{x'_D \in X'_D} \delta(f(x'_D) - g(x'_D))^2 + \Omega(g) \tag{7.2}$$

Here, $g$ signifies a linear model from a set $G$ of potential models. The optimization endeavor seeks to uphold local fidelity to the complex model $f$ while simultaneously ensuring the simplicity of the surrogate model $g$ by employing the complexity measure $\Omega$. The term $\sum_{x'_D \in X'_D} \delta(f(x'_D) - g(x'_D))^2$ quantifies the weighted sum of squared discrepancies between the predictions of the complex model $f$ and those of the linear model $g$ across perturbed instances $X'_D$. The weighting factor $\delta$ is linked to the proximity measure $\pi_x$ envisaged in the original formulation, expressing the distance of $x'$ to $x$.

To compute $\xi_D(x)$, we consider the set of perturbed instances $X'D$, each deriving from the image $x$ and altered along the dimension $D$ by the perturbation function 7.3.2. These instances form the basis of the input for the optimization problem proposed by LIME. The outcome of this optimization, the sparse linear model $\xi_D(x)$, approximates the behavior of the complex model $f$ in the local space around the original instance $x$, relying on the features defined by the dimension $D$.

## 7.4 Application

The following describes the practical application of the explainer approach presented in Section 7.3 for your skin image classifier. In order to investigate the functionality of the explanation set, perturbed instances will be generated using sample data, and sparse local explanations will be generated using a DNN classification model and an local interpretable model.

The application of the previously described implementation has been integrated into the use case described in Section 6.1. Specifically, this use case forms the basis for the selection of the dataset and the structure of the ML pipeline, with adjustments made to adapt the proposed explanation approach, which is initially described.

### 7.4.1 ML Pipeline Adaption and Experimental Setup

The foundation of this study was the HAM10000 dataset (Tschandl *et al.* 2018) as described in Section 6.1.1, which was utilized for the implementation of this research. To reduce the complexity of classifying skin lesion images, and enabling the applicability of the *ABCD-Rule*, an initial delimitation of the problem space was defined. This restriction involved limiting the selection of classes to two categories: Melanoma and Nevus. According to Table 6.1 on class descriptions, Nevus denotes benign neoplasms of melanocytes. In contrast, melanomas are typically characterized by their asymmetry concerning the distribution of color and structure, whereas melanomas are defined as malignant neoplasms that can manifest in various forms (Tschandl *et al.* 2018).

A second adaptation in the design of the ML pipeline was the modification of the model architecture. The narrowed data selection allowed for choosing a less complex model architecture, MobileNetV2 (Howard *et al.* 2017), which possesses fewer trainable parameters. Furthermore, the ML pipeline was significantly simplified by disabling data augmentation, thus limiting preprocessing steps to include scaling, as described in Section 6.1.2. This model architecture has been employed in several studies and has proven effective for this specific case of skin image analysis (Adegun *et al.* 2021).

| | Nevus | Malanoma | $\Sigma$ |
|---|---|---|---|
| # of Samples | 1,354 | 216 | 1,561 |
| True Positives | 1,150 | 144 | 1,294 |
| False Positives | 203 | 72 | 275 |
| $F_1$-Score | $\approx 0.91$ | $\approx 0.57$ | $\approx 0.74^*$ |

**Table 7.1:** Evaluation results of the trained skin image classifier based on the MobileNetV2 (Howard *et al.* 2017) architecture. To ensure that class imbalances have no influence, * 'macro' is specified as $F_1$ average strategy.

Limiting the scope to a binary classification problem resulted in a selective data base of 6,705 images for Nevus and 1,113 images for Melanoma (refer to Figure 6.2 for class distribution of HAM10000). Following the data splitting logic implemented in 6.1.2, this dataset was separated into training and test data in an 80/20 ratio, after which the tailored skin image classifier was trained to distinguish between nevus and melanomas. The predictive performance achieved on the test data can be observed in Table 7.1, which, compared to the original implementation of the classifier with seven classes (refer to Table 6.3), demonstrates enhanced predictive accuracy, as measured by the $F_1$-Score, for the Melanoma class.

To circumvent technical challenges associated with segmenting lesion areas in dermatoscopic images, segmentation data associated with the dermatoscopic images from Tschandl *et al.* 2020, provided in hand created form, was incorporated into the realization of the perturbation logic. Including segmentation data enables targeted perturbations strictly within lesion areas in this setup, without necessitating further segmentation logic.

### 7.4.2 Hypothesis

The simplification of the problem space to a two-class problem allows for a focused investigation of the effects of perturbations on the predictions of the trained model for classifying skin images containing Nevus *(nv)* or Melanoma *(mel)*. In this section, hypotheses are formulated based on the described implementation and the application of the explanatory approach.

The investigation includes medically relevant features. Positive perturbation $\delta^+$ is defined as altering the sample to more closely resemble a melanoma, whereas negative perturbation $\delta^-$ involves diminishing melanoma-like characteristics in the image. Medically irrelevant perturbations $\delta'$ neither remove nor introduce critical features. Defining $y$ as the probability of a given input sample $x$ belonging to a particular class, the study derives hypotheses about the black box model $f$ based on the aggregated predictions $\hat{y} = \frac{1}{n}\sum_{i=1}^{n} f(x')$, with $n$ perturbed inputs $x'$ related by $\sim_\delta$:

$Ha_1^{(nv)}$ Prediction for *Nevus* will **decrease** with **positive** perturbation:

$$Ha_1^{(nv)}(x, x', f) = \{x \sim_{\delta^+} x' \Rightarrow y > \hat{y}\}$$

$Ha_0^{(nv)}$ Prediction for *nevus* will increase or remain unchanged with positive perturbation.

$Hb_1^{(nv)}$ Prediction for *nevus* will **increase** with **negative** perturbation:

$$Hb_1^{(nv)}(x, x', f) = \{x \sim_{\delta^-} x' \Rightarrow y < \hat{y}\}$$

$Hb_0^{(nv)}$ Prediction for *nevus* will decrease or remain unchanged with negative perturbation.

The hypotheses apply universally across all medically relevant dimensions of perturbation. However, their validity is contingent on the specific input sample, rendering them not independent of the sample's true class. In the context of a two-class problem, such as the one being analyzed, the hypotheses for melanoma $\{Ha_1^{(mel)}, Ha_0^{(mel)}, Hb_1^{(mel)}, Hb_0^{(mel)}\}$ are applicable in a reversed formulation. This implies that negative perturbations should shift the prediction towards a nevus, while positive perturbations should incline the prediction towards melanoma.

Medically irrelevant dimensions should be independent of both the true class of the original image and the dimension to which the perturbations belong. We therefore hypothesize the following:

$Hc_1$ The black box model is **inherent** to medically irrelevant perturbations:

$$Hc_1(x, x', f) = \{x \sim_{\delta'} x' \Rightarrow f(x) = f(x')\}$$

$Hc_0$ Perturbation along medically irrelevant dimensions have significant effects on predictions.

To bridge the theoretical foundation laid out with practical insights, the following section presents a comprehensive analysis of the formulated hypotheses through experimental validation.

## 7.5 Empirical Results

To study the presented domain-specific explainer in more detail, model interpretations were generated from test samples, focusing on elucidating the explainer's performance by analyzing selected samples. High-confidence true positive cases and low-confidence false negative cases are illustrated in Figures 7.3 and 7.4. Another selection criterion was that a significant class flip manifests in at least one dimension.

Each figure is organized as follows: Dimension values are presented as headings, under which the maximally perturbed images are displayed. Additionally, for each sample, scatterplots are provided in each dimension. These plots correlate the black box model's prediction values, plotted on the ordinate, with the perturbation's intensity, depicted on the abscissa.

The prediction scale in all scatterplots is standardized to the range of $[0; 1]$, correlating with the class associated with the sample. The perturbation strength is, defined by the limits of $\delta$, scaled between $[-1; 1]$. Values in the negative range indicate negative perturbations, while positive range values correspond to positive perturbations.

Moreover, each scatter plot features a dashed vertical line at the zero position, demarcating the boundary between negative and positive perturbations. The classifier's prediction for the original, non-perturbed image, denoted by $f(x)$, is marked with a red cross on the $y$-axis. For every input sample, a total of $n = 50$ perturbed samples, $x'$, were generated, spanning both negative and positive directions. This approach allows for a comprehensive analysis of the explainer's effectiveness across various degrees of image perturbation, shedding light on the robustness and reliability of the model under study.

### 7.5.1 True Positives

The first case examines the model output behavior for correctly classified samples. The focus is on a domain-specific approach, aimed at addressing the question: „In which dimensions does the model retain its accuracy?"

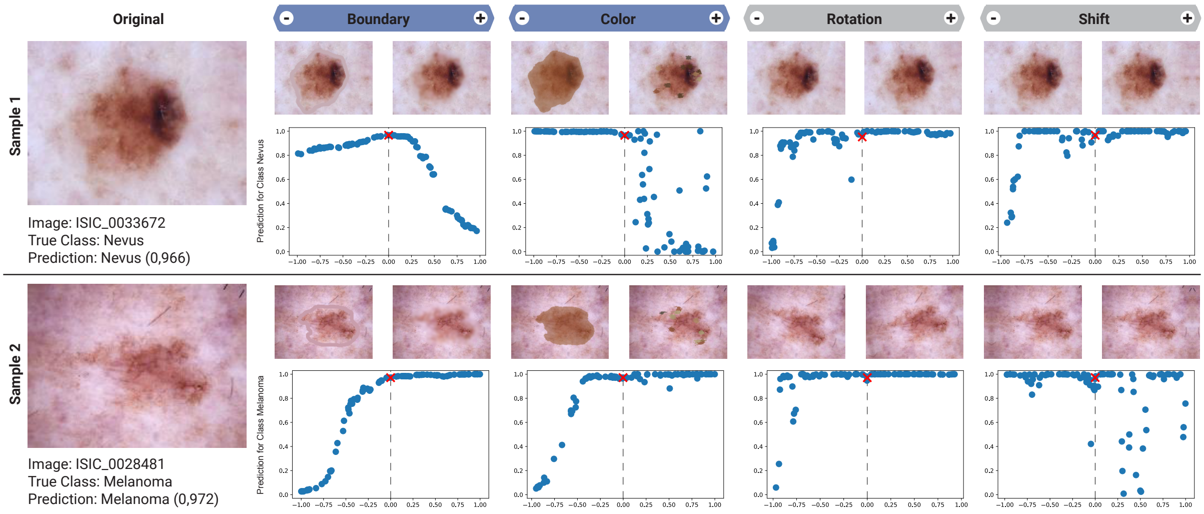


**Figure 7.3:** Overview of the observed model behavior. Two original samples, both **correctly** classified (**True Positives**), with their maximum perturbations for all four explanation-dimensions. Scatterplots under the perturbed images show the prediction of the black box model, each acquired along the indicated dimension.

This is explored using two selected samples illustrated in Figure 7.3 as a basis for hypothesis testing.

In the case of **Sample 1**, it is observed that the model's prediction within the medically relevant dimensions of *Boundary* and *Color* diminishes in the context of positive perturbation. This observation leads to the acceptance of hypothesis $Ha_1^{(nv)}$ and the rejection of $Ha_0^{(nv)}$. Conversely, for negative perturbation, hypothesis $Hb_1^{(nv)}$ is only accepted for the *Color* dimension, as the prediction remains constant. However, in the *Boundary* dimension, a different trend is noted: $Hb_1^{(nv)}$ must be rejected in favor of $Hb_0^{(nv)}$, since the prediction neither stagnates nor increases but decreases at a consistent rate.

The third hypothesis, labeled $Hc$, focuses on two medically irrelevant dimensions. Despite variations in the prediction at individual perturbation points, a general observation is made that the prediction level remains high in both positive and negative value ranges. The average prediction value across all perturbation values is $\hat{y} = 0.931(-0.035)$ for *Rotation* and $\hat{y} = 0.959(-0.007)$ for *Shift*, with parenthetical values indicating deviations from the prediction of the non-perturbed image. Given these minimal deviations, hypothesis $Hc_1$ is accepted and $Hc_0$ rejected.

For **Sample 2**, a reversal of the hypotheses' statements is necessitated, as the sample represents a melanoma case. Here, the prediction for positive perturbation is expected to either increase or remain constant, while for negative perturbation, a decrease is anticipated. This prediction behavior is confirmed in both medically relevant dimensions of *Boundary* and *Color*, leading to the acceptance of both $Ha_1^{(mel)}$ and $Hb_1^{(mel)}$ hypotheses.

Similar to **Sample 1**, fluctuations in the classifier's prediction along the perturbations are noted in medically irrelevant dimensions. Thus, the average prediction along all perturbation variables is recalibrated, yielding $\hat{y} = 0.971(-0.001)$ for *Rotation*, and $\hat{y} = 0.907(+0.065)$ for *Shift*. The deviation observed in the prediction allows for the acceptance of

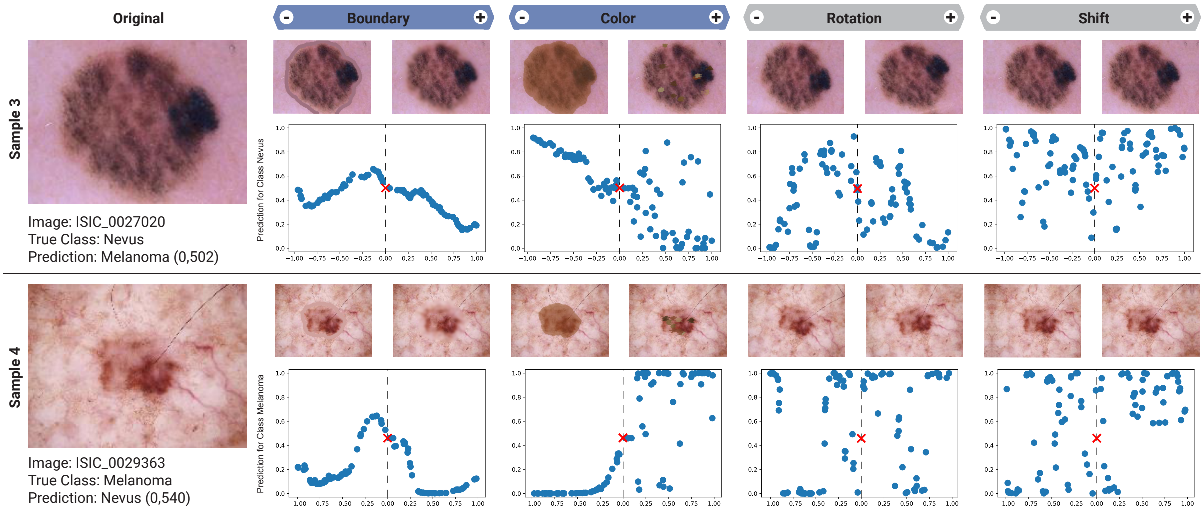


**Figure 7.4:** Overview of the observed model behavior. Two original samples, both **incorrectly** classified (**False Positives**), with their maximum perturbations for all four explanation-dimensions. Scatterplots under the perturbed images show the prediction of the black box model, each acquired along the indicated dimension.

hypothesis $Hc_1$ for *Rotation*, but does not support hypothesis $Hc_1$ for *Shift*, resulting in the acceptance of $Hc_0$.

### 7.5.2 False Negatives

The second case explores model explanations for incorrectly classified samples, with two test images exemplified in Figure 7.4. In this context, the aim is to elucidate the reasons behind the model's failure in classification.

In the positive direction for **Sample 3** within the *Boundary* dimension, hypothesis $Ha_1^{(nv)}$ is accepted, while $Hb_1^{(nv)}$ is rejected concerning the negative direction. Conversely, in the *Color* dimension, $Hb_1^{(nv)}$ is accepted. In this case, the prediction fluctuates above and below the expected values in the positive direction, thereby contravening the stipulations of $Ha_1^{(nv)}$. Nevertheless, this hypothesis is further accepted, considering that the average prediction in the positive direction is $\hat{y} = 0.318(-0.184)$, significantly lower than the prediction of the non-perturbed input image. When examining hypothesis $Hc$, it is immediately evident in both *Rotation* and *Shift* dimensions that $Hc_1$ must be rejected, as the classifier's prediction varies widely across all perturbation variables.

Regarding **Sample 4**, the analysis indicates that for the *Boundary* dimension, both hypotheses $Ha_1^{(mel)}$ and $Hb_1^{(mel)}$ must be rejected, as perturbations in both positive and negative directions lead to a decrease in prediction. The scenario differs for the *Color* dimension. Here, hypothesis $Ha_1^{(mel)}$ is initially accepted due to decreasing prediction values in the negative area. However, assessing the positive area is less straightforward, with values fluctuating. Despite this, given that the average prediction value $\hat{y} = 0.688(+0.148)$ is significantly higher than that of the non-perturbed sample, hypothesis $Hb_1^{(mel)}$ is accepted and $Hb_0^{(mel)}$ rejected.

Similar patterns are observed in the medically irrelevant dimensions for **Sample 3**, leading to the rejection of hypothesis $Hc_1$. The prediction behaviors do not exhibit a clear pattern, possibly indicating the model's weak performance for both samples. However, the output from the explainer remains valuable, providing insights into the model's lack of robustness and guiding improvements in both decision-making processes and model redevelopment.

### 7.5.3 Aggregation

To further explore the functionality and mechanisms of the domain-specific model interpretation method presented, this section introduces aggregated results from the broader dataset, in addition to the analysis of individual samples, as seen in Sections 7.5.1 and 7.5.2. By synthesizing interpretations across all test patterns, we aim to provide a macroscopic view of the predictions made by the trained model. All samples in the test set were subjected to the guided sampling procedure and, for all four Ⓑ, Ⓒ, Ⓡ, Ⓢ dimensions, sets of perturbed images were created in positive and negative expressions.

Subsequently, sparse linear explanations were generated, and the accumulated predictions of the black box model on these perturbed images were aggregated using Kernel Density Estimation (KDE). While the top row displays the results for the Nevus class, the bottom row represents the results for the Melanoma class. The resulting heatmaps encapsulate the normalized value ranges and offer, in Figure 7.5, an overview of the black box model's predictive behavior across the perturbation strength spectrum for the four dimensions.

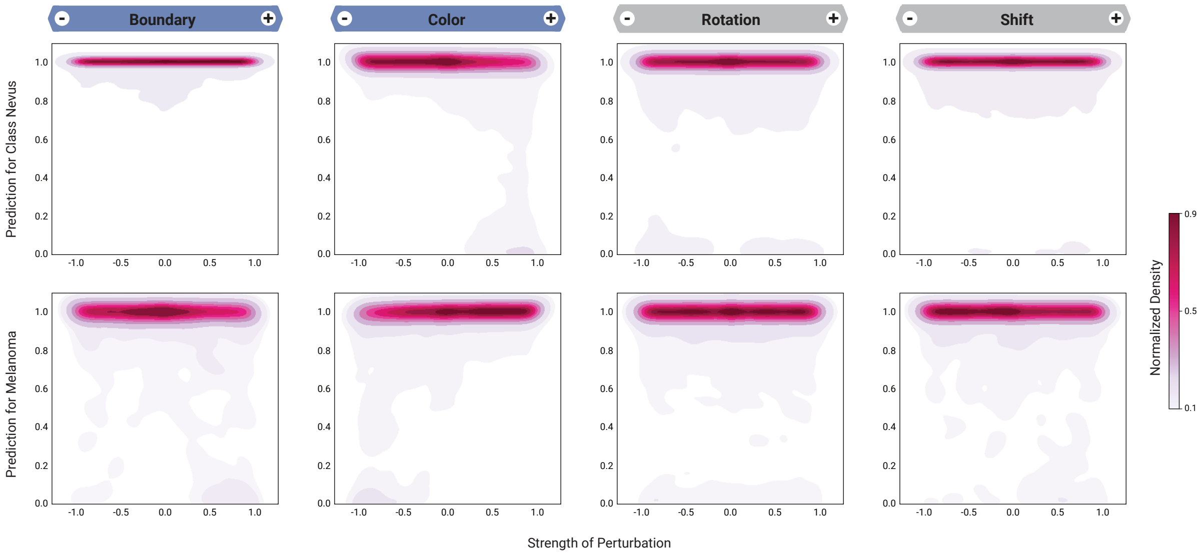


**Figure 7.5:** Aggregated model predictions for nevus and melanoma.

The intensity of the color in the heatmaps correlates with the density of predictions, with darker regions representing a higher concentration of prediction values. In other words, the representation corresponds to the overlaid scatter diagrams from Figures 7.3 and 7.4 for all test samples and have been transformed into heatmaps for better readability using KDE.

As observable in all eight depicted heatmaps, there is a clear, concentrated image of high density at high prediction values. In the medically relevant dimensions of Boundary and Color, a focused band of high density in the central region of perturbation strength ($\delta \approx 0 \pm 0.5$) is noticeable. Yet, while a variability is discernible, the heatmap for Nevus prediction values in the Boundary dimension shows an exception: Here, the model seems to lean towards high prediction values in both directions, hence in this dimension $Ha_1^{(nv)}$ and $Hb_0^{(nv)}$ are discarded, and $Ha_0^{(nv)}$ and $Hb_1^{(nv)}$ are accepted. The same applies to Melanoma in the reverse form.

In the Color dimension, the hypotheses for both classes $Ha_1^{(nv)}$, $Ha_1^{(mel)}$, $Hb_1^{(nv)}$, and $Hb_1^{(mel)}$ can be accepted. When the strength of perturbation reaches its limit values, the density dissipates, indicating variability in the model's predictions under more pronounced perturbations. It shows that predictions decrease with positive perturbation for Nevus, as well as with negative perturbation for Melanoma.

For the medically irrelevant dimensions of Rotation and Shift, the density plots exhibit a pronounced band of high density consistent with the perturbation strength range. This suggests that predictions are less sensitive to perturbations in these dimensions and maintain high reliability across the entire spectrum of perturbation intensity. Thus, we can accept $Hc_1$ and reject $Hc_0$.

The KDE plots provide a nuanced understanding of the model's behavior by aggregating the predictions of the black box model across all four dimensions and perturbation strengths. The density distributions in this experiment suggest that the trained black box model, after domain-specific interpretation, appears to consider correct attributes for the prediction decision in the Color dimension. The attributes associated with the Boundary dimension do not seem to be of crucial importance for predicting the class of Nevus. However, the visible prediction fluctuations with perturbation in medically irrelevant dimensions seem to indicate that some samples in the test set lie in borderline areas.

## 7.6 Discussion

Integrating domain-specific knowledge into a model interpretation method, as demonstrated by implementing the adjusted perturbation logic by LIME analogous to the ABCD rule provided by dermatologists, can pave the way for the accessible interpretability of ML models for people. In particular, specialized audiences such as doctors, but also patients, can benefit from such approaches by bridging the gap between technical complexity and practical applicability.

The empirical results presented in the study offer insights into the response behaviors of the black-box model to perturbed images, allowing hypotheses to be made about which features might have been important for the respective predictions. However, it is crucial to translate this information into a form understandable to the user. One possibility is to use bar charts for visual representation instead of scatterplots, highlighting the importance of individual attributes according to the ABCD rule, thus enabling a more intuitive interpretation of the results.

The observations on model behavior were based on a single model instance. It should be noted that results can vary depending on the model architecture and the training datasets used. The transferability of the specific changes in the model's predictions due to perturbations on input images to different diseases and medical imaging modalities is limited. The effectiveness of the adapted method is closely linked to the quality and representativeness of the training data used.

Furthermore, the application of the method was conducted using already existing lesion segmentations. In real-world scenarios, such segmentations are often unavailable and would have to be manually provided by an expert. The possibility of automated segmentation of the image areas by th ML model exists but introduces an additional step in the processing pipeline, which may be fraught with uncertainties. Furthermore, it is essential to reflect on the methodological approach where the perturbation of the input samples occurs successive, while the simultaneous perturbation of multiple dimensions remains to be investigated.

Another significant aspect in analyzing the empirical results, both qualitatively and quantitatively through aggregated values, is the lack of direct evidence for the correlation between the observed importance of a feature dimension and its actual value according to the ABCD rule. To bridge this gap, it appears necessary to develop and implement additional metrics that can comprehensively assess the effectiveness and accuracy of the explanatory approach. This could create a more solid basis for interpreting ML model predictions in the context of specific medical guidelines, ultimately contributing to improved patient care.

## 7.7 Summary

The skin image classifier in a CDSS can serve as a second opinion for a dermatologist. To a certain extent, the research community has already made it possible to realize such tasks today. However, the models implemented in these AI-based systems are limited to generating predictions without allowing the dermatologist to question the rationale behind the classifier's decision-making process. XAI-Methods and interpretable ML techniques aim to bridge this gap.

The adaptation of LIME for the specific domain of skin image classification, with a focus on differentiating between melanoma and nevus, illustrates a domain-specific approach for local explainability. Thanks to the model-agnostic architecture of LIME, this approach is suitable for DNNs as well as for other types of models. A hypothetical scenario in a decision support system could be that a dermatologist identifies a lesion as a nevus while the model classifies it as melanoma. This discrepancy leaves the treating physician (and the patient) in a skeptical position, raising the question: „Why?“ The presented domain-specific approach could answer that harmonizing the color in the lesion reduces confidence in the classifier's prediction. The doctor might recognize a color inconsistency in the dermatoscopic image that is not visible on the lesion itself and thus be able to explain why the classifier erroneously tended towards a diagnosis of melanoma.

A CDSS either confirms the doctor's diagnosis or suggests a contrary diagnosis. In both cases, it proves to be highly beneficial if interpretable, human-understandable explanations for the predictions can be generated. Domain-specifically adapted approaches foster increased trust and a deeper understanding of the system.

Following the current results from Section 7.5, future work could investigate the two remaining medically relevant dimensions, asymmetry and differential structure, for which the work of Barata *et al.* 2019 provides an overview of feature extraction in dermoscopy image analysis. Furthermore, Ali *et al.* 2020 demonstrate a method to extract these features from lesions. In line with the findings of Almaraz-Damian *et al.* 2020, another potential research endeavor could be to use the data modified by our explanation component as training data. This would allow for the evaluation of the resulting model's performance and check if the predictions follow a different pattern. Moreover, the approach of perturbation-based explanations, considering both medically relevant and irrelevant features for diagnosis, may potentially be applied to other medical fields.

# 8 Stakeholder-oriented Model Interpretations

**8.1 XAI-Compass: A Conceptual Guide for Stakeholder Engagement . . . 204**
8.1.1 Goals of Model Interpretations . . . . . . . . . . . 205
8.1.2 Model Interpretations across the Life Cycle . . 206
8.1.3 Stakeholder Groups of Model Interpretations . 207
8.1.4 Touchpoints for Model Interpretations . . . . . . 209
**8.2 Case Study . . . . . . . . 210**
8.2.1 Interpretable Machine Learning for ECG Signal Classification . . . . . . . 210
8.2.2 Implementation . . . . . 212
**8.3 Stakeholder Engagement 215**
8.3.1 Domain Expert Interviews . . . . . . . . . . . . 215
8.3.2 Scenario-based Focus Group Discussions . . . 218
8.3.3 Survey at Workshop . . 222
8.3.4 Discussion and Recap . 225
**8.4 Summary . . . . . . . . . 226**

The significance of interpretability in AI systems is increasingly recognized as a human-centric issue, given the pivotal role these systems play across various domains of life and their escalating complexity. This realization has led to a multidisciplinary approach that has prioritized the integration of social science insights with technical advancements in AI to enhance system transparency and user trust. (Dhanorkar *et al.* 2021)

Similarly, Miller 2019 and Wang *et al.* 2019 advocate for the design of AI systems to mirror human thought processes to enable clearer and more meaningful explanations. Against this backdrop, Miller 2019 proposes the paradigm of viewing the field of XAI through a socio-technical lens, highlighting the inseparability of technical systems from the diverse social aspects that come into play during their application. In his paper, where explanation and explicability are further examined from philosophy, psychology, and cognitive sciences perspectives, he contends that AI explanations are fundamentally social and evolve through interaction, emphasizing the need to recognize their dynamic nature to facilitate real-world applications.

A socio-technical perspective suggests that the need for interpretations of model outputs - and further, the need for explanations - arises from interactions between various stakeholders and the AI system itself (Wolf 2019), as well as extends beyond the direct users of the system to include additional participants. For instance, Gilpin *et al.* 2018 emphasize the importance of aligning model interpretations with human semantic concepts to ensure that explanations are technically accurate as well as comprehensible and relevant to users' real-life experiences and knowledge frameworks.

Therefore, integrating model interpretations is interdisciplinary, requiring various stakeholders' involvement in implementing IML methods. This chapter introduces an integrative approach to enhance explainability in developing trustworthy AI. A structured framework incorporates diverse stakeholder perspectives into IML implementation, optimizing accessibility to model interpretations. After introducing this concept in Section 8.1, Section 8.2 showcases a case study on the technical implementation of model interpretations for the ECG signal classifier discussed in Section 6.2. This case study forms the basis for the practical application of the concept in real-life scenarios. Finally, in Section 8.3, the integration of various stakeholders into the use case is presented. Three methods for integrating different groups are presented and applied, and the results are analysed to gain insights into the different perspectives of model interpretation and trust in the AI system.

Excerpts of this chapter, such as the XAI-Compass (Section 8.1) and the results of the focus group discussions (Section 8.3.2), were published as part of the proposed SocioTechXAI Integration Framework (STXIF) in Ziethmann *et al.* 2024.

## 8.1 XAI-Compass: A Conceptual Guide for Stakeholder Engagement

In order to create a conceptual framework for stakeholder engagement in the field of IML, this section introduces the XAI-Compass, visually depicted in Figure 8.1. The XAI-Compass is designed as a navigational tool to guide developers and researchers through the complex landscape of stakeholder needs and expectations regarding model interpretability. It complements the AL development life cycle proposed in Chapter 3 by integrating aspects of interpretable ML, thereby enriching the framework with a focus on explainability throughout the phases of the AI systems development.

This instrument structures the multifaceted aspects of stakeholder engagement within the domain of IML. It is based on a 3x3 matrix categorizing roles, phases, and objectives, as proposed initially by Gleicher 2016 and later revised by Hong *et al.* 2020 through survey work in the context of IML, with the organizational chart in Figure 8.1 recognizable by its distinctive color palette and use of bold formatting to highlight key terminology.

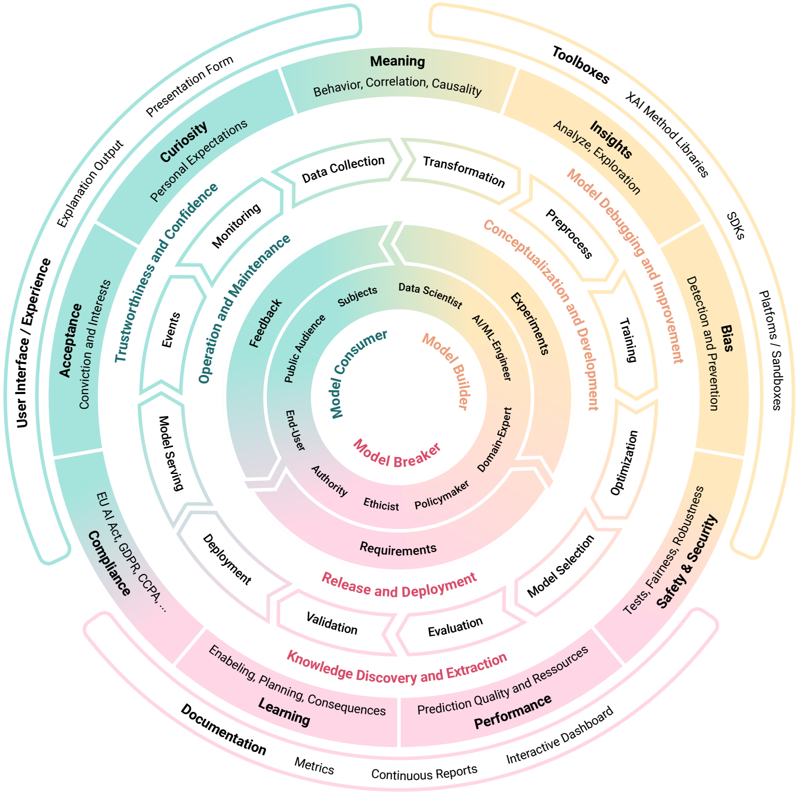


**Figure 8.1:** The XAI-Compass, following the 3x3 matrix by Hong *et al.* 2020, offers a holistic view of stakeholder groups, AI life cycle stages, and objectives. The concentric layers illustrate the dynamic interplay and iterative nature of the roles, phases, and goals within the ecosystem of IML. Following the color-coding scheme, three possible touchpoints are listed on the exterior.

The outer layer outlines various objectives pursued through model interpretations, categorized into three core goals (cf. Section 8.1.1). The intermediate layer details the AI life cycle's phases, segmented into Conceptualization and Development, Release and Deployment, as well as Operation and Maintenance (cf. Section 8.1.2). At the core of the XAI-Compass, the diagram delineates three primary stakeholder groups, namely Model Builder, Model Breaker, and Model Consumer, each associated with specific roles in the XAI process (cf. Section 8.1.3).

The XAI-Compass abstracts and synthesizes the prevailing discourse in the XAI research field, aiming to bridge the technical and the multidisciplinary dimensions of integrating IML into AI projects. It provides a pragmatic framework to address the foundational 'W-Questions' related to model interpretations as outlined by Arrieta *et al.* 2020 - *Who* is involved, *What* activities are taking place, *When* do these activities occur, and *Why* are specific interpretability objectives being pursued? Following Gleicher 2016 and Hong *et al.* 2020, these questions are directly linked to the 3x3 structure of the XAI-Compss: *Who*: Roles, *What* and *When*: Phases, and *Why*: Goals.

Addressing these questions is crucial for both comprehending stakeholder perspectives and advancing the development of tailored explanations, thereby encouraging a more profound contemplation of the various roles that actors play within the model interpretation narrative. Furthermore, the XAI-Compass allows for the derivation of potential touchpoints for stakeholders in various phases, targeting the question of *How?*. These touchpoints, and which approaches can be utilized, will be illuminated in Section 8.1.4.

However, the objectives of the stakeholders regarding model interpretation will first be examined.

### 8.1.1 Goals of Model Interpretations

The outermost layer of the XAI-Compass comprises the goals that are pursued using methods of the IML. Here, technical methods converge with user-aspired goals pursued with an explanation. These goals are categorized according to Model Debugging and Improvement, Knowledge Discovery and Extraction, and Trustworthiness and Confidence.

In the overarching goal of Model Debugging and Improvement, the objective is to optimize ML models in terms of their predictive performance and functional accuracy. IML methods are utilized to gain a deep understanding of the model mechanics, which enables comprehensive investigation and analysis (Mohseni *et al.* 2021). This application of the methods has an inspecting and diagnostic character, meaning that the models are dissected with the aim of understanding how inputs are processed into predictions, thereby paving the way for precise improvements (Bhatt *et al.* 2020; Hong *et al.* 2020). In addition to improving predictive performance, a key element of this goal is the identification and circumvention of biases (Amann *et al.* 2020). Here, IML techniques are crucial as they are capable of highlighting biases and using the insights gained to correct unwanted (unfair) biases in the data or the model in a subsequent step. Another focus is on the safety and reliability of the models, where interpretation tools can be used to conduct tests to evaluate the model for fairness and robustness and possibly to identify vulnerabilities (Dey *et al.* 2022).

Knowledge Discovery and Extraction is the second key goal. Often, due to the vast volumes of data, not the data itself but the trained model is used as a source of knowledge, with interpretation techniques being able to derive new knowledge (Markus *et al.* 2021; Liao *et al.* 2022). Another application is the facilitation of learning, where these tools open new pathways for education and assist in predicting and understanding outcomes (Meske *et al.* 2022). Furthermore, compliance, such as with the EU AI Act (European Commission 2024), is another goal, which is why methods of IML are used to meet legal requirements as well as industry-specific standards regarding the requirements of explainability.

Trustworthiness and Confidence, as the third main goal, underscores the significance of IML techniques in the context of trustworthy AI. By providing model interpretations, the aim is to establish and enhance acceptance based on the assurance that an ML model's capabilities align with the stakeholders' interests (Mohseni *et al.* 2021; Laato *et al.* 2022). Interpretability allows a sober view of AI decision-making and thus harmonizes system operations with stakeholder expectations. Curiosity is driven by the user's aim to understand the workings of an AI system. In other words, they pursue the goal of reconciling the outputs of the ML model with their own mental model of the world (Miller 2019; Markus *et al.* 2021). While the methods of interpretability are not directly capable of this, they can contribute to revealing the meaningfulness of model outputs, thereby sharpening stakeholders' perception of correlations and causalities.

### 8.1.2 Model Interpretations across the Life Cycle

The traditional phases of the AI life cycle are presented at the middle level, aim to answer the 'What' and 'When' questions. Despite their differences across various domains and use cases, they are generally divided into three overarching phases: conceptualization and Development, Release and Deployment, and Operation and Maintenance. The XAI-Compass captures the iterative essence of these phases, illustrating how stakeholders are typically assigned to these stages. Between the role and phase layers, the feedback loops run counterclockwise, encompassing experiments, requirements, and feedback.

In an AI system's Conceptualization and Development phase, fundamental decisions are already made that affect model interpretations, such as planning the model architecture. Here, iterative activities that are analogous to the conceptual phases of the AL development life cycle (cf. Chapter 3) take place, extending from data engineering to modeling. In this phase, there is often preemptive collaboration with Model Breakers and Model Consumers to establish a common understanding that facilitates later adoption and trust. In this regard, the primarily active Model Builders conduct experiments in which results of model predictions are generated and communicated to the relevant parties (Kaur *et al.* 2020).

During the Release and Deployment phase, both prototypes and new model versions are frequently examined using interpretation techniques focused on instances, features, and model comparisons. Here, validation strategies may include creating test cases with Model Breakers to ensure that the models behave as expected, particularly in scenarios representing edge cases where the models might exhibit uncertainty (Bhatt *et al.* 2020). According to the results of the interview-study published by Hong *et al.* 2020, in this stage, the reliability of a model and the need to gain stakeholders' trust are of utmost importance.

In the phase of Operation and Maintenance, the focus of model interpretation shifts to ensuring the ongoing reliability and relevance of the model in the deployed application. This phase is characterized by the predominant involvement of Model Consumers, which is why this step of interpretability is more about understanding the system behavior during direct interaction with new data and use cases. Because non-experts are primarily involved in this phase, a user-oriented and practical approach to model interpretations is necessary (Ferreira *et al.* 2020; Laato *et al.* 2022).

### 8.1.3 Stakeholder Groups of Model Interpretations

In the work of Langer *et al.* 2021 and Meske *et al.* 2022, various approaches to categorizing stakeholders and their respective objectives for an model interpretation were proposed. The XAI-Compass adheres to the categorization by Hong *et al.* 2020 and recognizes three primary stakeholder groups within the model interpretability sphere: Model Consumers, Model Builders, and Model Breakers. These form the core of the Compass, describing the diverse actors related to an AI system.

This delineation is not merely categorical but further functional, shedding light on each stakeholder's diverse responsibilities and contributions to an AI system. It is an intentional simplification that streamlines the complexity of stakeholder engagements into a focused analysis of the most pertinent phases and goals relative to their roles. Moreover, it reflects the dynamic nature of stakeholder positions, acknowledging the fluidity and overlap between these roles. For instance, individuals may not be confined to a single role; rather, they may transition among roles depending on the context and objectives at hand, as exemplified by the color gradients for the entities of a stakeholder group in Figure 8.1.

The listed roles refer to typical terms in the literature. Starting with the group of Model Builders, approximately representative entities provide an excerpt from the technical roles defined in Section 3.4.2. These roles are specifically defined for the AL development life cycle but are equally relevant for AI projects in general.

**Model Builder.** This term refers to a group of professionals, such as Data Scientists and AI/ML Engineers, who are primarily involved in constructing AI systems. These individuals are tasked with designing, developing, and integrating ML models into existing systems, processes, and infrastructure (Hong *et al.* 2020). Their role is typically technical, and they possess a deep understanding of AI systems and the underlying concepts of ML models, encapsulating a group that Mohseni *et al.* 2021 categorize into XAI target users „AI Experts“ and „Data Experts“, yet within the concept of the XAI-Compass, they are collectively referred to as Model Builders. Their expertise thus includes, for instance, the functionalities and techniques for improving ML models, as well as the associated knowledge of data preparation activities and the learning processes of algorithms (Suresh *et al.* 2021).

A significant part of a Model Builder's tasks, which usually manifest as iterative, involves the initial development and refinement of AI systems, including debugging and improvement processes (Bhatt *et al.* 2020). Therefore, they are directly involved in the implementation of the models. Typically, these stakeholders operate close to the data, ML models, and further components of an AI system, requiring deep technical proficiency in analyzing complex problems and solving them through coding. Their direct access to various tools and technologies, along with their specialized training, enables them to conduct detailed investigations of the AI system and make adjustments.

**Model Breaker.** This group comprises a range of individuals who distinguish themselves from Model Builders by their role in evaluating, validating, and testing AI systems for their alignment with specific goals and compliance standards. This includes domain experts, product/project managers (Liao *et al.* 2020), decision-makers (Wang *et al.* 2019; Meske *et al.* 2022; Laato *et al.* 2022), and compliance officers, including, for instance, auditors, regulators, and policymakers as well as ethicists (Hong *et al.* 2020). Unlike Model Builders, who work closely with data and ML models, Model Breakers typically do not engage directly with the technical aspects, illustrating their non-technical role within the stakeholder group. This contrast becomes more apparent when considering the non-technical roles of the provided AL development life cycle outlined in Section 3.4.1, whose descriptions seamlessly integrate into the attributes of this group. This leads to a shift in the skill set towards understanding broader implications and application contexts of AI. Although they may not possess the deep technical AI knowledge that the Model Builder stakeholder group does, their expertise is crucial for assessing whether an ML model achieves its intended objectives through correct functionality.

Model Breakers have the critical task of acting as a secondary authority to identify discrepancies or problems within AI systems. They essentially provide a necessary counterbalance to Model Builders and offer perspectives based on practical domain-specific applications, ethical considerations, and legal frameworks for AI systems. Their focus on the broader operational, ethical, and regulatory contexts means that, while their work is indispensable for the integrity and compliance of AI systems, it involves less direct interaction with the data and models. In this role, they are responsible for ensuring that the ML models are technically sound and align with the overarching goals of the AI system, the project, or the organization and that they meet the communicated requirements.(Hong *et al.* 2020)

**Model Consumer.** This stakeholder group is defined by individuals who are the end-users of AI systems and those affected by their decisions (Meske *et al.* 2022). It includes a wide range of professionals across various fields who use AI for decision support, such as hiring, loan approvals, and medical diagnoses. The group further extends to those directly impacted by these decisions, such as job applicants, bank customers, or patients (Hong *et al.* 2020). In addition, this group includes a broader category of stakeholders - public audiences - XAI target users that Mohseni *et al.* 2021 refers to as novices.

In contrast to Model Builders and Model Breakers, Model Consumers typically lack an in-depth understanding of the underlying technologies in AI systems. A unique aspect of this group is that their access to data and models is often limited and generally confined to a superficial, results-oriented level to ensure necessary accessibility for those who usually do not have separate tools at their disposal (Laato *et al.* 2022). Therefore, their engagement with AI systems primarily revolves around the interpretation of outcomes and the applicability of these outputs to their needs and contexts.

### 8.1.4 Touchpoints for Model Interpretations

Integrating model interpretations into AI systems introduces distinct touchpoints for stakeholder groups at various life cycle phases. To effectively engage all stakeholder groups, it's crucial to tailor the application of interpretation tools and techniques to these touchpoints, aligning them with the roles and phases to aid in achieving objectives and facilitating tailored access and interaction with interpretation methods.

Derived from the XAI-Compass depicted in Figure 8.1, a critical touchpoint for Model Builders emerges during the Conceptualization and Development phase. As noted, this group possesses deep technical knowledge, enabling them to select and apply suitable methods close to the data and model. Several renowned methods have been encapsulated in software packages for integration and adaptation during development.

As Schwalbe *et al.* 2023 indicate, a single interpretation method often falls short in elucidating all relevant model aspects, leading to adopting tool collections that amalgamate multiple interpretation methods within a unified library. Their review highlights several toolboxes, with popular open-source projects including IBM's *AI Explainability 360* Toolbox (Arya *et al.* 2019), *InterpretML* (Nori *et al.* 2019), and *Alibi* (Klaise *et al.* 2021). These toolboxes provide an SDK-like usability, simplifying access to various interpretation techniques by encapsulating the supported methods and catering to a diverse technical audience, from beginners to advanced data scientists and AI/ML engineers.

Another potential touchpoint derived from the XAI-Compass in the Release and Deployment phase involves Model Breakers. These individuals might lack the technical capability to access toolboxes yet are keen on analyzing models using suitable interpretation methods. Besides model metrics, interpretations at the instance level in counterfactual scenarios can play a role in examining model behavior and aligning it with intended goals during this phase.

Reports serve as an appropriate medium, offering a formal touchpoint for Model Breakers and a low-threshold access in the development and conceptualization phase, embodying not just reactive but proactive objectives as well. Utilizing the CI/CD/CT Runner concepts introduced in Section 4.2.3, for instance, enables automated and parameterized execution of routines that process implemented interpretation methods within the ML pipeline, presenting the generated outputs in a report format. This approach further suits computation-intensive tasks unsuitable for interactive interpretation, like aggregating multiple interpretation results across a large dataset.

An additional touchpoint with model interpretations arises in the Operation and Maintenance phase, involving Model Consumers. In this phase, where interaction with the AI system and model occurs in real-world scenarios, ensuring accessibility to potential model interpretations for a lay audience is crucial to facilitate context-specific use. This necessitates considering the presentation form in designing the user experience and UI, whether through visually prepared information e.g., throuh heatmaps, text, or rule-based explanations.

Besides integrating into the end-user interface, where the underlying technology and interpretation methodology might be abstracted depending on the use case, interactive dashboards offer the potential as an adequate tool for other stakeholder groups at this touchpoint. Alongside proprietary solutions, such as those in the *Azure* cloud plattform (Microsoft 2024), popular open-source projects include *OmniXAI* (Yang *et al.* 2022), *ExplainerDashboard* (Dijk *et al.* 2023), and *Shapash* (MAIF Data Scientists 2020). These interfaces enable a dynamic and interactive selection of interpretation methods and data instances, proving especially beneficial for participants wishing to present complex matters intuitively. They provide a platform where users can explore algorithms and data instances, analyze them, and offer feedback in an explorative manner.

## 8.2 Case Study

In the following section, a case study is introduced that utilizes the setup presented in Section 6.2 of a 12-lead ECG signal classification model for detecting arrhythmias. By employing methods of IML, this case study serves as a foundation for examining stakeholder integration, presented in Section 8.3. Before delving into the application of the algorithms, a brief overview of the current state of research will be provided.

### 8.2.1 Interpretable Machine Learning for ECG Signal Classification

In recent years, the area of IML for AI-supported ECG signal data analysis has gained significant attention for its application in CDSS. In numerous studies, IML methods have been applied to various model architectures for different application purposes.

LIME (Ribeiro *et al.* 2016), as a model-agnostic method, has spread in this domain as a popular technique. In the context of ECG data analysis, Abdullah *et al.* 2023 have further developed LIME for domain-specific implementation. They propose „B-LIME“, a process that uses bootstrapping to consider the temporal dependencies of the QRS complexes in the signal data.

Chen *et al.* 2023 implement LIME within a hybrid framework consisting of a one-dimensional convolutional neural network fused with a Support Vector Machine to detect hypertension in ECG data. The LIME results indicate which specific ECG waveform features contribute to differentiating between normotensive and hypertensive patients, providing valuable interpretative assistance primarily tailored for medical professionals.

Hughes *et al.* 2021 present an application of LIME to a CNN. The authors trained a model to predict 38 diagnostic classes from 12-lead ECG data. They leave the pathological relevance of the marked areas open but point out that the main advantage of their method lies in processing large amounts of data. As the authors mentioned, the combination with LIME could reveal novel ECG correlates by highlighting unexpected LIME segments, possibly indicating as yet unknown disease mechanisms.

In addition to local explanations, Ivaturi *et al.* 2021 pursue a dual perspective by applying LIME, offering global explanations for the problem of atrial fibrillation detection from ECG signals. They implement this by dividing the ECG signals into fixed-length segments and then aggregating the weights generated by LIME over several samples to infer which signal area is relevant for predicting atrial fibrillation.

The authors in Rouhi *et al.* 2021 apply not only LIME but further SHAP (Lundberg *et al.* 2017) to their ML models, which were trained to classify atrial fibrillation in ECG signals. The effectiveness of the SHAP technique has been elucidated as an enhancement to performance. The results were used to select features, which were then utilized in a new ML model, which achieved higher predictive performance than the original ML model.

Furthermore, Jekova *et al.* 2022 use SHAP to analyze atrioventricular synchronization as a diagnostic criterion for atrial fibrillation, with results indicating that heart rate variability and specific ECG markers for atrioventricular conduction times significantly influence the predictions of their ML models.

Angelaki *et al.* 2021 apply SHAP to analyze the importance and interaction of clinical parameters and electrocardiographic features in detecting abnormal left ventricular geometry. They focused on a single class in their work and determined the importance of parameters such as age, blood pressure, and various ECG parameters for the diagnosis. Similarly, the authors Ibrahim *et al.* 2020 focus on predicting a specific class and show how Shapley values can be used to identify the features that contribute most to the classification of myocardial infarction using XGBoost.

The work by Neves *et al.* 2021 is innovative in incorporating temporal dependency into ECG time series to make the classification of heartbeats more explainable using SHAP. In contrast, Agrawal *et al.* 2022 present a one-dimensional CNN classifying ECG signals from post-COVID and healthy subjects. They use SHAP to interpret the relevant ECG sections and conduct the analysis on both a patient-wise and lead-wise basis.

Furthermore, Zhang *et al.* 2021 conduct interpretation with SHAP at the level of individual patients, examining their ML model for classifying cardiac arrhythmias from 12-lead ECG signals. They extend perspectives and perform an analysis with SHAP at the population level as well, allowing the authors to quantify the influence strength of individual features on the model predictions, thus obtaining a more global view of the model.

### 8.2.2 Implementation

The model for classifying cardiac arrhythmias, as described in Section 6.2.2, was utilized to facilitate the case study. Implementing two IML methods, LIME and SHAP, was intended to generate various model interpretations.

**LIME.** For the use of LIME, Model Builders have access to a dedicated open-source Python library[1]. However, natively, it does not provide the functionality for perturbation of signal data,which led to an expansion of the logic during its integration into the ML pipeline described in Section 6.2.2 at the validation stage.

Following the approach by Ivaturi *et al.* 2021, the logic initially divides a given ECG signal into segments of fixed, configurable length to account for the temporal dimension. A random selection of a given number of signal segments was implemented across the entire ECG signal.

While Ivaturi *et al.* 2021 describe the application to single-lead signal data, an extension was added for a cross-lead selection of signal segments. This approach followed the sliding window principle, capturing the temporal dependency across the twelve leads by simultaneously selecting segment sections across all twelve ECG leads.

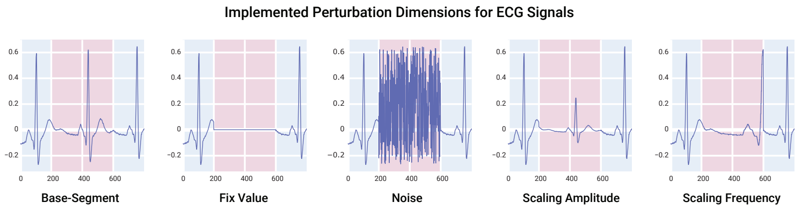


**Figure 8.2:** Perturbation dimensions for LIME signal data for sampling for local exploration in LIME. The sample (original signal) is divided into segments, which are then perturbed along the predetermined strategies. The figure shows the exemplary selected base segment (pink background) in its original form, as well as the output after perturbation along respective dimensions.

In line with the concept from Chapter 7, domain-specific perturbation dimensions, as depicted in Figure 8.2, were implemented. The figure shows an excerpt of an unprocessed raw ECG signal with 800 sampling points from *Lead I*, corresponding to a duration of 2 seconds at an underlying sampling rate of $400Hz$.

Highlighted in pink is an exemplary selection of a 400-sampling point-wide segment that spans one second. Besides the base signal, four different signal perturbations are recognizable:

- **Fixed Value:** Assignment of a specific value to the selected segment, which in this example (here = 0) corresponds to the scenario of asystole for the marked area.
- **Noise:** Set to random values within the segment, simulating electronic or sensor noise.
- **Scaling Amplitude:** Changing the values by a given factor, either decreasing (reduction of amplitude swing, as shown) or increasing (enlargement of amplitude swing).
- **Scaling Frequency:** Doubling the values (stretched frequency, as shown, which artificially „widens“ the area) or reducing the values (compressed frequency, which artificially „narrows“ the area).

[1] See https://github.com/marcotcr/lime, last accessed on 2024-04-02.

**SHAP.** SHAP served as alternative method of interpretation, with an open-source library[2] available to Model Builders for this purpose. In alignment with the application of SHAP within the domain of ECG signal classification proposed by Zhang *et al.* 2021, Shapley values were computed, using Expected Gradients as proposed by Sundararajan *et al.* 2017, at the signal level („Patient-Level" in the terminology of Zhang *et al.* 2021) to determine the relevance of individual sample points within the ECG signal, as observed in Figure 8.3.

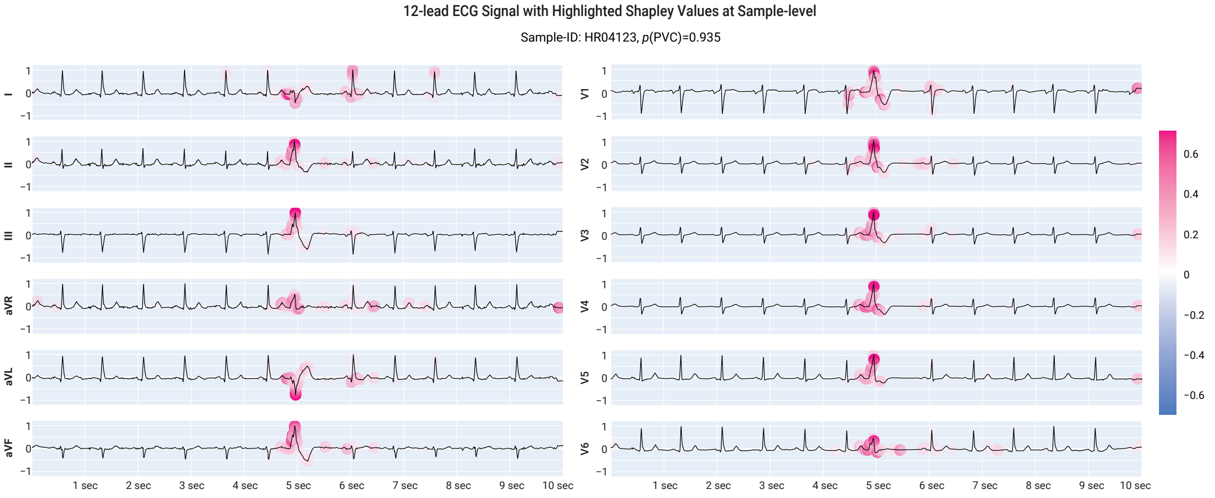


**Figure 8.3:** 12-lead ECG signal with highlighted shapley values at sample-level. Displayed are normalized signal values of the twelve leads over 10 seconds. The pink-colored overlay points on the ECG signal mark an increased Shapley value, indicating areas where specific signal segments are relevant for the models' (correct) prediction for class „PVC".

Here, the twelve leads are labeled with their respective lead designations, and for clarity, the signal values correspond to preprocessed, normalized ranges. The x-axis delineates the time axis in seconds.

For the selected sample (ID: HR04123 from the PTB-XL dataset, cf. Section 6.2.1), which the trained model predicted to be the correct „PVC" class with a probability of $p \approx 0.935$, regions around the 5-second mark are significantly highlighted, showcasing characteristic „PVC" spikes. These areas, marked in various shades of pink, indicate positive Shapley values supporting the „PVC" classification. Conversely, the areas depicted in blue represent negative Shapley values that counter the „PVC" classification. This local observation aligns with the experiments presented by Zhang *et al.* 2021, who demonstrated similar findings for another „PVC" class sample in their setup.

The implementation extended the calculation of lead-wise Shapley values to ascertain the relevance of individual ECG leads. The lead-specific Shapley values calculated for the sample depicted in Figure 8.3 are visible on the left side of Figure 8.4. This visualization elucidates the significant influence of leads V2 and V4 on the model's predictive accuracy, whereas the influence of leads V1 and V6 is minimal.

Furthermore, determining lead relevance and the direct additivity of Shapley values facilitate the aggregation of values across the entire test dataset, a property exploited by Zhang *et al.* 2021, who refer to this in their work as „Population-Level Interpretation". The result of this aggregation

[2] See `https://github.com/shap/shap`, last accessed on 2024-04-02.

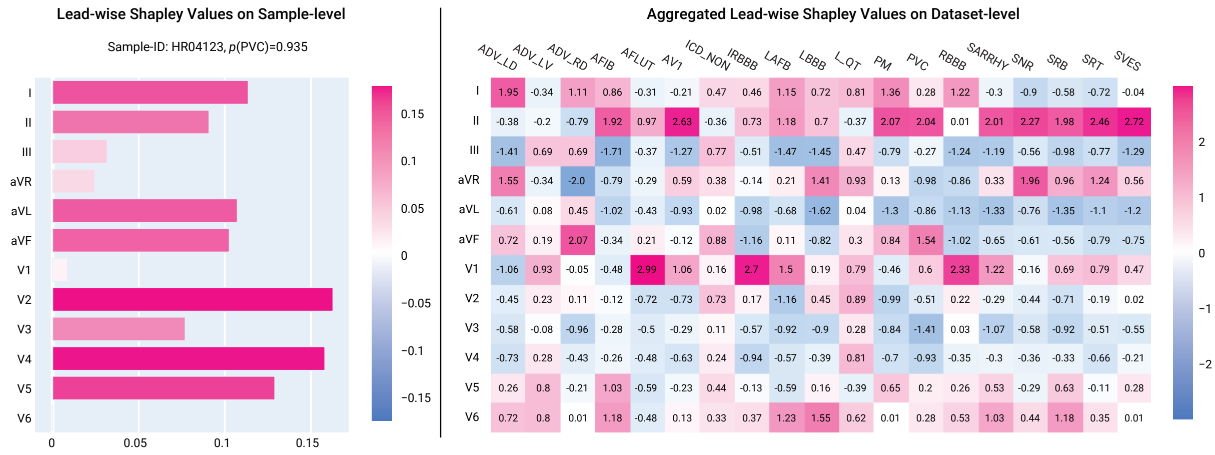


**Figure 8.4:** Lead-wise Shapley values for 12-lead ECG signals. The left chart provides a lead-wise breakdown for a single sample (local), illustrating the relevance of each lead to the model's prediction for the „PVC" class. The right panel aggregates Shapley values across the entire test dataset for each lead, enabling a more global model interpretation.

of Shapley values for each true-positive predicted sample on the test dataset is shown on the right side of Figure 8.4.

The matrix clarifies which leads were significant in various classes. The same color scheme was selected, meaning that positive Shapley values (depicted in pink) suggest that a particular lead contributes positively to the model's prediction. In contrast, negative values (in blue) indicate a contrary effect - the more intense the color, the stronger the contribution to the prediction outcome.

As evidenced by the chart, particularly lead II demonstrates high positive Shapley values across several classes, suggesting that the signal in this lead contains key indicators for the model. Other leads, such as aVL and V1, exhibit mixed or neutral significance. However, on a class-by-class basis (columns), it is notable that for classes „AFLUT", „IRBBB", and „RBBB"[3], lead V1 displays high Shapley values. This observation is partially consistent with the findings of Zhang *et al.* 2021, whose model does not cover all these classes but whose „Population-Level Interpretation" seeks to verify the correlation with clinical procedures of ECG readings, identifying that this lead exhibits characteristic features of these classes.

As recently seen, the implementation of both methods, LIME and SHAP, enables the creation of various interpretations, which can be utilized for subsequent studies involving stakeholder engagement.

[3] For terms of abbreviations please refer to Table 6.6

## 8.3 Stakeholder Engagement

The following section delves into stakeholder engagement within the context of an AI-based ECG classifier, building on the case study introduced earlier. The focus is on exploring the different perspectives of various stakeholder groups as identified in the XAI-Compass.

By conducting three distinct studies, the aim is to gather feedback from these stakeholders to gain a comprehensive understanding of their views on the interpretability of the model and how it influences their perception of trustworthiness. The insights gathered will contribute to a deeper understanding of how different stakeholders perceive and evaluate the proposed model interpretations.

### 8.3.1 Domain Expert Interviews

Including domain experts, primarily in their role as part of the stakeholder group of **Model Breakers**, forms the foundation for ensuring trustworthy AI systems, especially in terms of their explainability. In this context, expert interviews with subject-matter experts represent an appropriate qualitative method that bridges the gap between technical innovation and practical application by providing insights into specific requirements as well as the relevance of the generated model interpretations.

#### Study Setup

The study setup focuses on conducting semi-structured interviews with two heart surgeons from the German Heart Center at the Technical University of Munich who specialize in analyzing ECGs. The participants, due to their involvement in an AI research project, were familiar enough to provide informed statements on the use of model interpretations in the application context of an ECG signal classifier.

The interviews were conducted remotely with both participants separately, utilizing the same prepared presentation slides as a basis for discussion in both interviews. The 90-minute interview was divided into three main parts:

1) **Visual Representations:** Participants were shown various visual representations that had been generated in advance and are displayed in Figure 8.5. The representation forms included two established and two innovative formats:
    a. *Values Highlighting:* color marking of signal points, as implemented by Agrawal *et al.* 2022 and Zhang *et al.* 2021
    b. *Background Heatmap:* color marking as a heatmap, as used by Abdullah *et al.* 2023, Hughes *et al.* 2021, and Van De Leur *et al.* 2022
    c. *Coloured Signal:* direct coloring of the signal line
    d. *Bounding Box:* insert bounding boxes in relevant areas of the signal

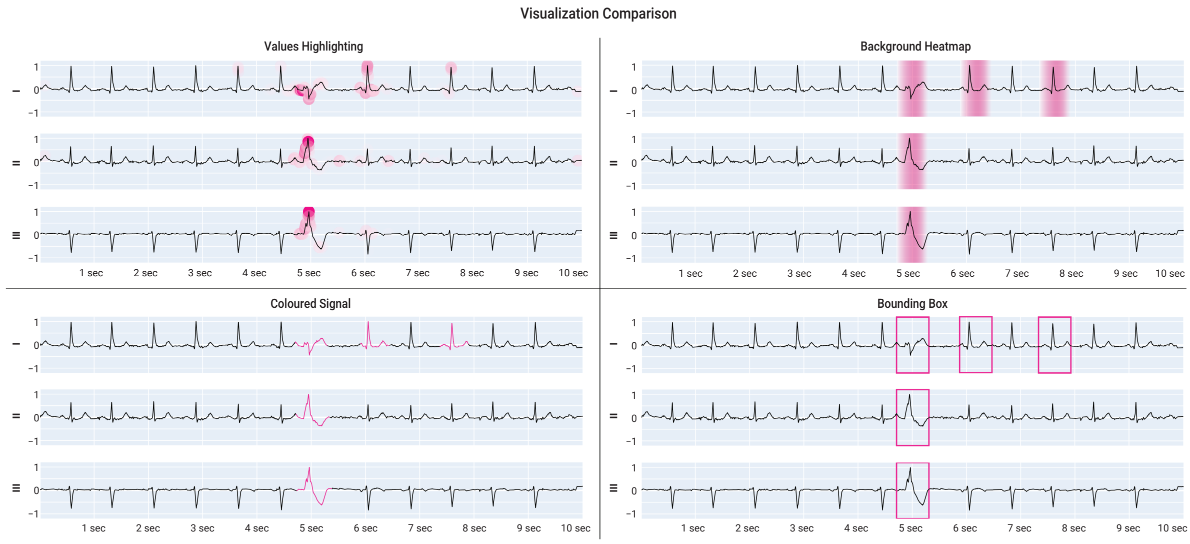


**Figure 8.5:** Comparative visualization techniques for local ECG classification model interpretations. The image shows four subgraphs, each with leads I, II and III of an ECG signal. Within these subgraphs, the results of the model interpretation are displayed in different formats. The colour-coded annotations indicate regions within the signal that contribute to the predictive decision process of the model.

Figure 8.5 shows these representation forms in comparison for an easier overview only for the first three leads. However, the participants were shown the representations for a 12-lead ECG, as shown in Figure 8.3. Various colors are not depicted, although different colors were shown during the interviews.

2) **Evaluation of Generated Model Interpretations:** In line with the taxonomy for the evaluation of explanations as proposed by Doshi-Velez *et al.* 2017, a human-grounded evaluation was conducted on the interpretations generated by the AI model. To this end, participants were presented with various examples of outcomes produced by the implemented IML methods. These outcomes included both the visual representations crafted by the interpretation algorithms and textual information about the model's output, such as predicted classes and confidence levels. Participants were tasked with assessing these outcomes for their medical significance and were invited to articulate their qualitative feedback through the „Think Aloud“ method (Lewis 1982).
3) **Feedback on Domain-specific Implementation:** Interviewees were given a brief lecture on the functionalities of the implemented IML algorithms LIME and SHAP. Subsequently, the domain-specific adaptations of these algorithms were explained. Participants were then asked to provide feedback on these methods, focusing on the intuitiveness and relevance, as well as any suggestions for improving the domain-specific implementation.

### Results

The thematic analysis of semi-structured interviews offered insights into the perception of model interpretations within the context of ECG signal classification. Both participants acknowledged the relevance of model interpretations for this proxy task of an AI-based CDSS, albeit prioritizing different goals as identified by the XAI-Compass. While one prioritized „safety“ as the highest goal, viewing „curiosity“ as less critical but potentially more relevant for patients, the other highlighted „analysis“ as the top priority in using interpretation mechanisms.

**Visual Representation.** The heart surgeons provided valuable insights from the domain experts’ perspective by assessing various visual representations of model interpretations. Discussions converged on a consensus that clarity and simplicity in visual representations, as in other systems, would enhance acceptance in daily practice. Both participants expressed concerns about the complexity and potential confusion caused by excessive colors and visual cues.

One participant noted the effectiveness of coloring only relevant areas and leaving the rest white, reminiscent of using highlighters and pencils on physical documents - a bridge between traditional and digital notation methods familiar to many clinicians. Bounding boxes intuitively resembled common practices of circling relevant findings during examinations. Another participant mentioned this form of representation reminded them of their medical studies, where textbooks outlined relevant areas with rectangular boxes.

Another favored method was the bounding boxes, appreciated for their clarity in delineating relevant sections. However, concerns were voiced that these boxes might obscure parts of the signal, suggesting a need for careful implementation to ensure all relevant data remains visible. The participant suggested overlaying the signal over bounding boxes to maintain ECG data readability.

Regarding the proxy task of using AI-supported ECG analyses in emergencies, both participants preferred simplicity, noting the heightened urgency of decision-making. Surgeons preferred visual cues like value highlighting and bounding boxes over background coloring in such scenarios. While providing a „clean“ appearance, the colored line of the signal might demand more attention in stressful situations, conflicting with the requirement to identify important information quickly.

**Evaluation of Generated Model Interpretations.** A recurring theme in evaluating generated model interpretations for selected samples was the importance of aligning AI explanations with established medical knowledge and diagnostic criteria. The feedback revealed varying reactions to the results, including cases where model predictions matched or differed from the diagnosis.

One participant pointed out the challenge of interpreting model predictions when some clinically relevant criteria were marked but not consistently correctly. Correctly colored areas would enhance trust in the model, while marking irrelevant areas could confuse. For correct model predictions (confirmed by the surgeons in the presented ECG signals), both noted that the markings from the model interpretations helped strengthen the understanding of the model’s decision-making process. However, incorrect predictions reduced trust, even if the color markings on the signal offered some insight into the error.

The output of aggregated lead relevance on both sample and dataset levels was deemed of limited informational value by one participant, considering it unhelpful to analyze an entire lead or multiple input data due to its imprecision. The other participant attributed limited importance to this model interpretation, viewing it from a verifying perspective as an additional but not particularly useful information source for model verification.

**Feedback on Domain-specific Implementation.** The intuitiveness of interpretation algorithms plays a crucial role in the explainability of AI systems. One participant found the explanations provided by LIME very intuitive, while both considered SHAP very intuitive, suggesting a preference for approaches that align with clinical reasoning.

Both participants reacted positively when discussing domain-specific adaptations of the implemented IML algorithms. Concerns were raised about implementing the perturbation logic „Noise" in LIME. For example, comparing the effects of perturbations could inadvertently add diagnostic features indicative of ventricular fibrillation. Both mentioned they would welcome selectable perturbation strategies, with one adding that choosing the perturbation segment manually for targeted manipulation to test model functions would be useful.

Using SHAP, both participants independently expressed interest in comparing similar classes and different classes, finding the current random selection of samples for Shapley value computation less medically relevant. They proposed that calculating feature relevance with samples of the same class would allow a more focused analysis while selecting different classes would further provide a medically relevant perspective. One participant suggested comparing samples classified as arrhythmias with those of normal sinus rhythm for a comprehensive analysis.

### 8.3.2 Scenario-based Focus Group Discussions

Focus groups, as described by Kitzinger 1995, are in-depth, open-ended group discussions where specific questions about a particular topic are explored, lasting one to two hours. They are seen as a form of group interviews, where the interaction among participants is emphasized as a key element, and open conversations are encouraged, allowing participants to express their opinions and introduce new ideas.

Based on the traditional concept of focus group discussions, scenario-based focus groups extend this methodology by incorporating specific, hypothetical, or realistic scenarios. These scenarios serve as stimuli for the conversation, helping to place participants in concrete, often more complex situations, which facilitates the generation of specific, application-oriented insights (Krueger *et al.* 2014).

Scenario-based focus groups expand and enrich the discussion space, as participants share their general views and respond to specific challenges and situations. They are widely used in qualitative research across various fields, including XAI, where they aid in understanding complex technologies through realistic scenarios (Johs *et al.* 2021; Zacharias *et al.* 2022; Bienefeld *et al.* 2023).

### Study Setup

In conducting the scenario-based focus group discussions, attention was centered on two stakeholder groups: **Model Builders** and **Model Consumers**. Two trios of participants were established, each representing the specified stakeholder groups. Separate online workshops were conducted with each group, starting with an introduction to the topic and methodology presentation.

The group of Model Consumers comprised medical students, while the group of Model Builders were AI Engineers with comprehensive data science knowledge. Participant group descriptions and self-assessment data on AI and ECG knowledge gathered from a questionnaire are detailed in Table 8.1. Medical students uniformly reported a moderate level of ECG knowledge captured by a categorical variable. In contrast, AI Engineers indicated having little to no experience in this area. These reports align with expectations, as were their self-assessments of AI knowledge, specified on a scale of 1 (lowest) to 10 (highest), where medical students rated their knowledge significantly lower than AI Engineers.

**Table 8.1:** Overview of participants in the scenario-based focus group discussions.

| Stakeholder Group | **Model Consumer** | **Model Builder** |
|---|---|---|
| Participants | Medical Students | AI-Engineers |
| Average Age | 25.33 | 42 |
| ECG Experience | Moderate | Little to None |
| Average AI Knowledge | 3.33 | 8.33 |

The context was set to the (hypothetical) use of an AI-based CDSS for diagnosing 12-lead ECGs to specify the discussion's use case, derived from the case study presented in Section 8.2. It's assumed that a trained ML model is an integral part of an ECG device analyzing ECG recordings to make arrhythmia diagnosis recommendations by outputting the likelihood of corresponding classes.

**Ground Scenario.** The ground scenario envisioned the clinical application of the AI-based CDSS, outlining a workflow where the device is used to record a 12-lead ECG, and the recording is analyzed using an ML model to output a diagnostic prediction through rhythm classification. An ECG sample, as shown in Figure 8.3, classified as „PVC" by the ML model, was used as a working example. The recorded ECG signal and the CDSS-produced „PVC" diagnosis were provided without additional data or explanations. When confronted with this scenario, the moderator invited participants to evaluate the CDSS's utility based solely on the information provided, asking, „What are your thoughts on the presented AI-based 12-lead ECG device?"

**Escalation Step 1.** Following the initial ground scenario discussion, the study moved to the first escalation level, building upon the base scenario. A novel element introduced here was the inclusion of a probability value in the ECG display immediately following the diagnosis, indicating the AI's diagnostic confidence and thus providing a variable explanation. Secondly, a new layer of visual explanation was added by incorporating the color codings shown in Figure 8.3, which resulted from calculated

Shapley values, into the presented ECG signal. This visual interpretation, created with the SHAP algorithm, offered localized clarification for the specific ECG sample used in the scenario. Participants were encouraged to contemplate and discuss the potential of this additional information layer with the question, „What do you think about this extension?“

**Escalation Step 2.** Transitioning to the second escalation level, the provided information was further expanded by disclosing details on the training dataset, including the number of „PVC“ class samples available for training and presenting five similar ECG samples of the „PVC“ class along with their Shapley values. Additionally, Shapley values at the lead level were visualized as bar charts and provided for aggregating Shapley values across classes, as shown in Figure 8.4. These details aimed to offer a comprehensive overview of the model output, with participants again encouraged to discuss with the question, „What do you think about this extension?“

### Results

The transcriptions were subsequently subjected to exploratory content analysis and examined based on the focal points of visual representation, explainability, trust, and suggestions for improvement. The evaluation revealed differentiated views of the two stakeholder groups, with the core statements from the discussions of both groups compared for overview in Table 8.2.

**Model Builders Perspective.** The group of AI engineers expressed concerns about the lack of critical information and emphasized the importance of providing additional data such as precision and secondary classifications, besides the probability of a diagnosis. Despite lacking in-depth ECG knowledge, they criticized using a normalized scale for representing ECG leads, as it obscures diagnostically relevant details like units of measurement and amplitudes of characteristic peaks.

In the first escalation step, the Model Builders valued the color highlighting of decision-relevant areas as applicable since these areas matched the signal sections they identified as conspicuous. In the second escalation step, they discussed the risk of information overload, especially from a patient’s perspective, who could be overwhelmed by it.

Nevertheless, they desired deeper insights into the training processes and more detailed information on the training data and model architecture. Regarding trust, they were initially skeptical about the model’s reliability due to the limited amount of data presented. They discussed factors that could strengthen their trust, such as compliance with industry standards and certifications.

**Model Consumers Perspective.** The group of medical students pointed out that comprehensive and accurate diagnosis requires additional medical information that was missing from the presented visualizations. They emphasized the importance of receiving further relevant ECG data such as RR intervals[4], alongside the model’s prediction.

[4] The RR interval is the distance between two R-peaks in the signal,

**Table 8.2:** Comparative summary of feedback from Model Consumers and Model Builders. This table presents a categorized compilation of key statements points from Model Consumers and Model Builders on representation, explainability, trust, and suggestions. The contrasting perspectives highlight areas of consensus and divergence between the two groups regarding the use of a CDSS and the provided explanations.

| | **Model Consumer** | **Model Builder** |
|---|---|---|
| **Representation** | [R1] ECG not fully displayed<br>[R2] Additional medical data (e.g., RR interval) required<br>[R3] Unfamiliar ECG format<br>[R4] Contain partly irrelevant information | [R5] Missing technical details<br>[R6] Indication of probabilities for secondary classification<br>[R7] Desire for marked signal areas<br>[R8] Confusion about normalised scales<br>[R9] Sample-ID irrelevant |
| **Explainability** | [E1] Need for explanation only in case of uncertainty or inconsitence with CDSS<br>[E2] Visualitation partly distracting<br>[E3] Partial information overload<br>[E4] Colouring is unimportant for simple diagnsoes<br>[E5] Indication of probability of diagnosis is important | [E6] Interest in more performance metrics<br>[E7] Highlighting relevant segments helpful<br>[E8] Improved information content through visual highlighting<br>[E9] Discrepancy in meaning of Shapley values<br>[E10] Amount of information potentially overwhelming |
| **Trust** | [T1] Emphasis on own ECG competence<br>[T2] Scepticism towards dependence on CDSS diagnoses<br>[T3] Knowledge of data quality is crucial | [T4] Trust increases with additional information<br>[T5] Visualisations that confirm own observations increase confidence<br>[T6] Need for medical knowledge to understand ECG patterns<br>[T7] Greater confidence when the obvious abnormality is recognised |
| **Suggestions** | [S1] Avoidance of abbreviations („PVC“)<br>[S2] Colour coding only for pathological sections<br>[S3] Red should be reserved as color for serious diagnoses<br>[S4] IML output could be suitable for educational purposes<br>[S5] CDSS should alert if data quality is poor | [S6] Demand for additional performance metrics (e.g. $F_1$ score)<br>[S7] Provide information on model architecture and training procedure<br>[S8] Explicit labeling of probabilities |

In the first escalation step, they praised the introduction of a probability assessment, which clarified the reliability of the diagnosis. In the second escalation step, they considered additional information from the CDSS output as potentially confusing. They recognized the value for professionals but highlighted the risk of misunderstandings without adequate medical knowledge. They advocated for solid medical education's importance in correctly interpreting the data.

Both groups showed interest in model interpretation, illustrating how the CDSS arrives at its diagnosis. Particularly in complex diagnostic cases, a detailed explanation was considered helpful in supporting the evaluation.

**Comparison of Perspectives.** When perspectives are compared, as shown in Table 8.2, it becomes apparent that both groups critiqued the unconventional representation of normalized ECG signals [R3,R8] and pointed out the irrelevance of certain data, such as the sample ID [R4, R9]. They agreed that information provided in the second escalation step could lead to information overload [E3,E10].

Differences were primarily observed from a technical versus medical viewpoint. While AI engineers focused on technical aspects, data representation, and model performance, demanding additional technical details to ensure the model's accuracy and reliability, medical students valued the information's medical accuracy and completeness. AI engineers saw XAI as a means to build trust regarding explainability and trust. In contrast, medical students considered explanations particularly important in the context of their studies and for assessing the CDSS's doubtful predictions, relying on their medical knowledge to evaluate the diagnosis.

These differences reflect the varied roles and fields of expertise, with Model Builders adopting the patient's perspective and primarily focusing on development, improvement, and usability, whereas Model Consumers highlighted the clinical utility and educational possibilities of the technology.

### 8.3.3 Survey at Workshop

To foster the involvement of a wider audience, a third methodology was integrated within the workshop on the topic of „Trustworthy AI in Medicine - from Efficient Data Annotation to Intuitive Model Explanation" (original German title: „Vertrauenswürdige KI in der Medizin - von effizienter Datenannotation bis intuitiver Modellerklärung"), which was conducted as part of the Science Congress in Ingolstadt 2022 (WIKOIN2022).

This consisted of conducting an interactive live survey in which participants were involved through structured questionnaires as part of the knowledge discovery process. This quantitatively oriented approach opens up an additional dimension by enabling the generation of overarching insights based on the aggregated feedback from participants. The workshop, which took place on-site and was openly accessible to the science congress visitors, was divided into three parts. While two of the three parts focused on AL and data annotation, the first part dedicated itself to the field of XAI.

#### Study Setup

The topic was introduced through a presentation contextualizing and showcasing the application example of the ECG classifier. This encompassed both the technical approach and the medical benefits of an ECG device. The integration of simplified ECG sensors into smartwatches was discussed to enhance the technology's accessibility for a broader audience. For illustration purposes, ECG signals of both normal heart rhythm and atrial fibrillation were presented, followed by a demonstration using the output of a professional 12-lead ECG device.

Subsequently, the workshop transitioned into an interactive hands-on session, inviting participants to partake in a live survey using their smartphones. This survey was facilitated by the freely available audience response system VoxVote[5], with participants gaining access by scanning a QR code. Questions, specifically tailored to the presented slides and initially probing the participants' professional backgrounds, were displayed.

[5] https://www.voxvote.com/

Mirroring the study design of the focus group discussions (as described in Section 8.3.2), the initial slide showed a 12-lead ECG depicting patterns of atrial fibrillation, supplemented by a hypothetical output from an ML model classified solely as „Atrial Fibrillation“ (*Ground Scenario*). The pivotal question posed to the audience was: „How much do you trust the result of the AI?“ Response options ranged from „Fully“, „Strong“, „Moderate“, and „Weak“ to „No Trust.“

An extension showed the same ECG signal, this time supplemented with a detailed output of the ML model. This provided deeper insight through the prediction probability for the primary diagnosis „Atrial Fibrillation - 99.09%“ and secondary classes „Normal Heart Rhythm - 0.02%“ and „Other - 0.89%“, complemented by probability values as a bar graph for verbal explanation (*Escalation Step 1*). A subsequent slide offered a visual interpretation of the model through color coding of the Shapley values, supplemented by information on the relevance of individual leads (*Escalation Step 2*).

The second survey focused on differentiating model interpretations. Based on previous expert interviews, various visualization forms were presented to a broad audience for clarity assessment. Participants were confronted with visualizations as depicted in Figure 8.5, with *Representation 1* „Coloured Signal“, *Representation 2* „Background Heatmap“, and *Representation 3* „Values Highlighting“ shown on separate slides, querying the understandability of each explanation. Response labels available were „Fully“, „Well“, „Moderate“, „Hardly“, and „Incomprehensible“.

Throughout the workshop, the presenter was available as a point of contact, allowing for responses to any question. Each question had an answering period of approximately one minute until the question was deactivated. Participants who did not respond were not excluded from the live survey and could participate in subsequent questions. Retroactive responses were not possible, as each question was only active for the duration of its answering period.

### Results

The 15 participants' data set generated during the live survey was captured using the Audience Response System, allowing for a comprehensive post-event analysis.

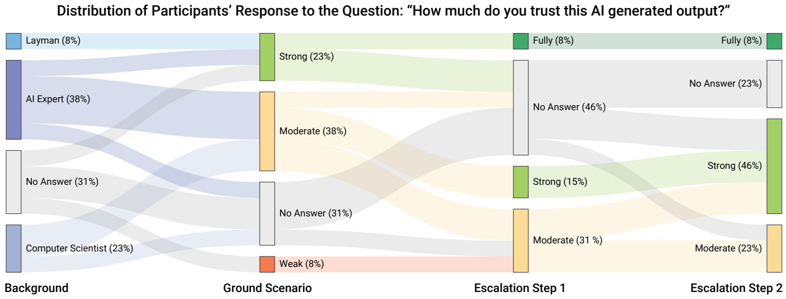


**Figure 8.6:** Trust in AI-generated output across escalation scenarios. This sankey diagram visualizes the distribution of participants' responses related to their trust levels in AI generated output, initially segmented by their professional background.

Figure 8.6 offers insight into the audience's responses concerning their level of trust in AI generated results, demonstrated through a 12-lead ECG classifier case study. The Sankey diagram illustrates the relative distribution of responses, categorized by the participants' professional backgrounds and sequenced according to the question order used in the workshop, starting with the baseline scenario and proceeding through two escalation steps.

In the Ground Scenario, a balanced distribution of responses among the categories „Strong,“ „Moderate,“ and a smaller proportion of „Weak“ was observed. As the scenarios escalated, the „Strong“ and „Moderate“ proportions diminished. Notably, in the second escalation step, there was a marked increase in the trust level deemed „Strong“. Moreover, it is noteworthy that in the Ground Scenario, the mere output of the class with the highest probability, without any additional information, resulted in some participants indicating „Weak“ trust. This category disappeared with the increment of information provided in the escalation steps, with a corresponding increase in the „Fully“ category. This shift indicates an increase in trust as more information is made available, with no migration towards lower levels of trust.

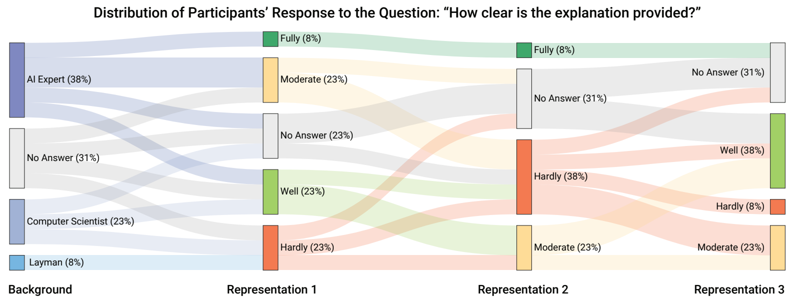


**Figure 8.7:** Survey result about clarity of provided explanations. The sankey diagram shows the participants' assessments of the clarity of the different representations provided, differentiated according to their professional backgrounds.

Figure 8.7 presents the distribution of participant responses regarding the explanations' clarity. The reactions for Representation 1 („Coloured Signal“) are heterogeneous, with the audience rating the clarity of the explanation as „Well,“ „Moderate,“ and „Hardly.“ Only a minority stated that the representation was „Fully“ clear.

The second form of presentation (Background Heatmap) seemed to polarize opinions, evidenced by a significant portion (38%) of participants rating it as „Hardly“ and none as „Well“, suggesting that Representation 2 was overall less comprehensible to the audience. Contrastingly, in Representation 3 (Values Highlighting), the proportion of „Hardly“ ratings was the lowest, and the proportion of „Well“ ratings the highest. Although no participants chose „Fully“, this representation appears to be the most understandable in comparison.

These findings must be viewed with respect to the expertise of the audience, a large segment of whom self-identified as AI experts or computer scientists. Given that the workshop took place at a scientific congress on the subject of AI, this is explainable. Another point of note is the significant portion of participants who did not provide any answers, possibly due to various reasons. This could reflect a certain degree of uncertainty on the part of the respondents or be due to the user-friendliness of the audience response system.

To conclude, the workshop results show that, taking into account the professional background of the audience, increased information provision leads to increased trust. The different representational forms of explanations were evaluated variably across all participant groups, with Representation 3, featuring feature highlighting, offering the most comprehensible approach.

### 8.3.4 Discussion and Recap

Reflecting on the studies conducted on stakeholder engagement, it becomes apparent that involving stakeholders from various disciplines provides a multiperspective view on the interpretations of the presented AI-based ECG classifier. However, it is crucial to critically consider that the context and selection of participants are decisive for the generalizability of the results.

The expert interviews described in Section 8.3.1, involving two cardiac surgeons, offer profound insights into the professional requirements and preferences regarding AI interpretations. Nonetheless, due to the limited number of participants, these interviews only represent part of the spectrum of opinions within the stakeholder group of Model Breakers. While these interviews provide valuable, practical perspectives, they highlight the limitations imposed by a small sample size.

Scenario-based focus group discussions, as outlined in Section 8.3.2 with the Model Builder and Model Consumer groups, illustrate these stakeholders' differentiated needs and perspectives. Despite the potential of this method, the small group size of only three participants should be noted, which falls below the ideal group size of four and eight people (Kitzinger 1995). Additionally, the online execution might impact the flow of conversation and group dynamics, potentially diminishing the authenticity and dynamism of the interactions.

The quantitative results of the survey conducted during a scientific congress on AI (Section 8.3.3) might have been influenced by the specialized audience. This context-specific participant selection and the workshop setting may not accurately reflect the views of a broader and more diverse audience.

Incorporating various stakeholders in these three studies, the investigation into the perceptions of interpretability of an AI-based ECG classifier offers the following insights:

- **Users prefer simplicity.** In particular, the medical experts favored clear and simple visual representations of model interpretations.
- **Intuition of visualization forms depends on the audience.** Visualization forms such as „Feature Highlighting“ were rated as understandable by both domain experts and other stakeholders. Additionally, „Bounding Boxes“ were familiar to medical professionals from their everyday experience.
- **Differing priorities in information content.** While AI engineers emphasize technical accuracy, medical students highlight the importance of medical relevance and the clarity of the information provided.
- **Increased trust with more information.** As more information was provided, higher levels of trust were observed across all studies.

- **Potential for information overload.** However, there is a risk of information overload, particularly when too many technical details are presented without adequate context.

## 8.4 Summary

This chapter has examined the integration of stakeholder perspectives in implementing IML. To this end, the XAI-Compass was introduced, a conceptual guide for stakeholder engagement that organizes a holistic view of stakeholder groups, AI lifecycle stages, and goals within the XAI ecosystem. It serves as a visual tool to structure the core questions about model interpretations: „Who?“, „When?“, and „Why?“. Additionally, various touchpoints were identified that aim to answer the question „How?“.

In a case study, the post-hoc XAI methods LIME and SHAP were adapted to a 12-lead ECG classifier, and corresponding visualizations were created. This provided insights into implementing further domain-specific model interpretations, focusing on three empirical studies involving stakeholder groups.

Using three different methods - expert interviews, scenario-based focus group discussions, and a survey conducted at a workshop - the perspectives and requirements of the stakeholder groups identified in the XAI-Compass, namely Model Builders, Model Breakers, and Model Consumers, were illustrated.

The results highlight the crucial need to align model interpretations and subsequent explanations with human-centered considerations such as understandability, relevance, and trust. Divergent perspectives may potentially create tensions, as evidenced by the varying preferences between medical professionals and technical participants. The trade-off between technical details and practical usability underscores the importance of stakeholder-oriented model interpretations.

The integration of stakeholder perspectives in the development and implementation of model interpretations is essential. The XAI-Compass provides a framework by identifying potential stakeholder groups, phases, and goals and proposes possible touchpoints. Engaging various stakeholders, as demonstrated in the conducted studies, results in improved acceptance and promotes access to relevant interpretations for all involved, aligning with the requirements for developing trustworthy AI systems.

# V. Conclusion and Outlook

# Conclusion and Outlook 9

Throughout this dissertation, the role of AL in the development of trustworthy AI was examined, introducing various approaches that enhance explainability and transparency in processes and systems. These approaches were evaluated in diverse ways and through various simulations and real-world studies involving human participants.

This final chapter summarizes and conclusively evaluates the three parts of this dissertation by revisiting the objectives introduced in Chapter 1 and recapitulating them with the findings of this work. Subsequently, an outlook on future research directions is given.

**9.1 Development in AL Projects . . . . . . . . . . . 229**
**9.2 Design of AL Systems . . 230**
**9.3 Interpretable Machine Learning . . . . . . . . . . 231**
**9.4 Future Work . . . . . . . . 232**

## 9.1 Development in AL Projects

This dissertation introduced a revised model for the development life cycle of AL and a technical workflow designed explicitly for data-centric AI projects applicable to AL projects. **Objective 1** aims to establish a uniform methodological foundation that enhances transparency and explainability at the process level in the execution of AL projects while harmonizing the diverse working methods of heterogeneous team members to promote efficient collaboration.

The presented AL development life cycle, as described in Chapter 3, organizes an AL project into phases and stages, providing a structured approach that merges best practices from traditional software development with the particularities of a data-centric project. The life cycle model is designed to be iterative and agile, accommodating the dynamic nature of the AL methodology, where data, models, and requirements can rapidly evolve due to new data and feedback loops. Technical and non-technical roles typical for an AL project are identified and defined, each responsible for a specific area of tasks.

A significant contribution of this part of the thesis is the adaptation of existing life cycle models for AI/ML projects to the AL methodology. The development life cycle for AL projects thus fills a gap by depicting a process model with conceptual phases and continuous iterations. This supports a team in maintaining coherence throughout the project duration and provides a framework that outlines *„what"* is to be done in each phase and stage of an AL project.

Building on this, several key concepts are introduced as part of the development methodology proposal presented in Chapter 4. These include principles related to data, code, and automation, which answer the question of *„how"* by presenting a practical guide for engineering teams. These principles are designed to be tool-agnostic, thus enabling application in various technical environments and use cases.

Implementing the proposed practices ensures that all changes in a project are traceable and transparent, addressing the requirements of trustworthy AI for transparency. This is achieved through a systematic tracking of artifacts throughout the life cycle, which provides a clear view of the origin and development of each component.

The effectiveness and relevance of the methodology were assessed in Section 4.5 across several dimensions: alignment with established best practices in SE, insights from expert interviews, and a meta-analysis of two project implementations. This comprehensive assessment has shown that the proposed methodology adheres to best practices and advances them by promoting a collaborative, agile, and data-centric development environment.

To conclude, **Objective 1** was successfully achieved by presenting a comprehensive model for the development life cycle and an engineering workflow that addresses the inherent challenges of AL projects and fosters a systematic, efficient, and transparent approach to implementing data-centric AI systems. This foundational methodology serves as a guide for development teams, empowering them to handle the challenges of data and model volatility while ensuring that the project remains aligned with the requirements for process transparency in the development of trustworthy AI.

## 9.2 Design of AL Systems

Chapters 5 and 6 explore an AL project from a systems perspective. **Objective 2** focuses on providing a framework equipped with essential components for effective data annotation as well as ensuring the end-to-end traceability of artifacts, to facilitate the implementation of AL systems.

The framework introduced in Chapter 5, LIFEDATA, integrates technologies for data provenance and artifact versioning. It consists primarily of two parts, each provided as open-source software. While the core framework, with its various interfaces and the relational database for storing annotations, forms the backbone, the generic project template includes a blueprint containing all key components relevant for executing an AL project with a focus on traceability. Both the core framework and the project template are agnostic to data/label types and model types, aiming to support a wide range of use cases.

The applicability of the LIFEDATA framework was demonstrated in Chapter 6 through two detailed use cases in the life sciences, each addressing unique challenges and showcasing the adaptability and effectiveness of the framework. Concepts from Chapter 4 were further applied here, illustrating the scalability of system components and infrastructure setup to meet the specific requirements of the use cases.

ML pipelines with DNNs were implemented for various data types, such as image and signal data, along with different querying strategies. The classification performance of the trained models was evaluated using established methods, and the effectiveness of the querying strategies was assessed through simulations using public, pre-annotated datasets. Moreover, in the use case involving ECG signal data, the applicability of the LIFEDATA framework as a tool for determining annotator consensus was demonstrated and analyzed in a study with human annotators.

The results presented in Part III indicate that the LIFEDATA framework is a practical solution for supporting development teams wishing to implement AL projects. The two parts of the framework contribute to achieving **Objective 2** by providing a blueprint for developers, integrating necessary technologies for data provenance and artifact versioning. Its modularity and data agnosticism support a variety of application scenarios and promote the implementation of AL across different domains. Thus, LIFEDATA contributes towards the realization of trustworthy AI systems with AL, by addressing the requirement for traceability.

## 9.3 Interpretable Machine Learning

The third part of this dissertation is dedicated to interpretable machine learning. In this context **Objective 3** aims to improve the accessibility of model interpretations through domain-specific adaptations and targeted alignment to the needs of different stakeholders. This addresses the requirements for trustworthy AI by explainability of systems and the need to make complex techniques understandable to diverse users.

As a contribution, a domain-specific interpretation algorithm was introduced in Chapter 7. This combines a human-understandable medical algorithm, the ABCD rule for classifying skin lesions, with a technical interpretation algorithm for ML models, LIME. This approach aims to bridge a domain-specific procedure in diagnosing skin lesions and a post-hoc IML-method to generate explanations that are relevant and accessible to end-users, especially medical professionals. The feasibility was demonstrated in a study where a DNN based classifier was trained to distinguish nevi and melanomas, thus fulfilling the domain-specific part of **Objective 3**. The generated results were analyzed using selected skin lesion images as well as in an aggregated form.

Chapter 8 deals with the integration of stakeholder perspectives into the model interpretations to address the stakeholder-specific adaption of **Objective 3**. The XAI Compass was introduced as a conceptual guide for stakeholder engagement in the field of IML, structuring stakeholder groups, phases, goals, and possible touchpoints. In a case study, concepts of domain-specific adaptations were transferred to the use case of ECG signal classification. The focus was on different levels of detail and forms of presentation of model interpretations. Three studies that used various interdisciplinary methodological approaches and targeted specific stakeholder groups examined the relevance of the model interpretations generated from the case study.

The integrative approach proposed in the dissertation is thus in line with **Objective 3**. By adapting model interpretation to the specific domain and involving various stakeholders, the accessibility and relevance of model interpretations are enhanced, which is crucial for meeting the requirements for the explainability of trustworthy AI systems.

## 9.4 Future Work

This dissertation has highlighted certain aspects of the role of AL in developing trustworthy AI that enhance the explainability and transparency of processes and systems, thereby opening up avenues for future research.

**Development in AL Projects.** In implementing AI systems with AL, adapting and expanding the AL development life cycle opens up numerous research opportunities and practical improvements for future projects. Applying the life cycle model to various use cases could offer the potential to explore specific roles and processes in different AL scenarios and enhance the life cycle model.

Similarly to well-known methodologies such as *Microsoft's* TDSP (Microsoft 2020) and CRISP-DM (Wirth *et al.* 2000), the AL development life cycle could be expanded into a comprehensive project workflow guide. This guide could define the process steps, potential tasks, and the necessary artifacts and reports for each phase. Another expansion involves implementing quality gates within the AL development life cycle. These quality gates would serve as checkpoints to ensure compliance with quality standards, which are particularly relevant in certifying AI systems.

To enhance the adaptability and efficiency of the proposed development methodology, future research could address challenges associated with the initial setup and high maintenance effort. This would involve developing concepts that simplify the technical and organizational setup and minimize the maintenance effort for the SE/AI engineering team. Such approaches include standardizing tools and automating setup processes, thereby enhancing the principles and workflow, e.g., around a platform development environment. Finally, there is a further need to expand the concepts of the development methodology, as revealed by the evaluation with best practices, which were not fully addressed in their presented form.

**Design of AL Systems.** The LIFEDATA framework introduced in this dissertation opens up a broad spectrum of future research directions. On the one hand, there are opportunities to apply LIFEDATA in a variety of additional use cases and domains, utilizing different data types, label types, and ML model types, which would require adapting the ML pipeline and other modules of the project template.

On the other hand, the addition and implementation of new, additional QSs could significantly expand the range of AL scenarios for LIFEDATA. New QSs, emerging from the specific requirements of different data types and ML model types, could also adapt or expand the hybrid approach integrated into LIFEDATA with mechanisms that, for example, alter the weighting of QSs in real-time to further enhance the efficiency of data annotation. Furthermore, future research could focus on the generation of pseudo labels that utilize and expand the semi-supervised learning implementation of LIFEDATA.

Another important area of research that could be enriched by the LIFEDATA framework is the investigation of inter- and intra-annotator agreements in additional contexts. This includes exploring the impact of mislabeled data, offering significant research potential that involves developing methods to handle faulty annotations. The LIFEDATA framework, for instance, could be enhanced with consensus-building strategies or verification and correction mechanisms, which could also contribute to the development of trustworthy AI systems with an AL loop.

Additional potential expansions could focus on the LIFEDATA project template, which can be supplemented with further components tailored explicitly to different - previously unsupported - AL scenarios, such as the stream-based approach. This would further increase the application spectrum of the framework.

**Interpretable Machine Learning.** IML and XAI remain central and expansive fields of research that will continue to gain importance in the future. Longo *et al.* 2024 describe in their manifesto the open challenges and future research directions in these areas, some of which relate to the concepts and subsections addressed in this dissertation.

New domain-specific interpretation methods could build on the approaches introduced in this work and contribute to improving the explainability of ML models in additional fields. For instance, the perturbation logic could be adapted to other medical disciplines (e.g., X-ray imaging), or the concepts could be transferred to other domains. An important direction for future research in this context is the development of interpretation methods for new types of ML models, which, in addition to the problem of image and signal data classification considered in this dissertation, could also include applications in language processing.

Meanwhile, the introduced interpretation algorithm for the skin image classifier offers room for expansions. It could be supplemented with additional dimensions not covered in the initial implementation and tested in further skin image classifiers.

Regarding stakeholder integration, the exploration of user-specific customization possibilities for interpretations and explanations is another area that justifies future research. By developing techniques that adapt interpretations to the needs and understanding of various stakeholder groups, the understanding, trust, and acceptance of AI systems can be increased. A systematic assessment of the effectiveness of interpretation methods in terms of decision-making in various application cases is required. Long-term studies that investigate the use of systems in real environments could provide valuable insights into the practical effectiveness and impacts of these technologies. Such studies could help bridge the gap between theoretical research and practical application, ensuring that IML methods are not only effective but also reliable in their context and contribute to the trustworthiness of AI systems.

# Bibliography

1. Abdullah, T. A. A., Zahid, M. S. M., Ali, W. & Hassan, S. U. B-LIME: An Improvement of LIME for Interpretable Deep Learning Classification of Cardiac Arrhythmia from ECG Signals. en. *Processes* **11** (2023) (cited on pages 210, 215).
2. Abraham, A. & Dreyfus-Schmidt, L. Cardinal, a metric-based Active learning framework. en. *Software Impacts* **12** (2022) (cited on pages 125, 126).
3. Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., Adam, M., Gertych, A. & Tan, R. S. A deep convolutional neural network model to classify heartbeats. en. *Computers in Biology and Medicine* **89,** 389–396 (2017) (cited on page 161).
4. Adadi, A. & Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* **6,** 52138–52160 (2018) (cited on page 43).
5. Adegun, A. & Viriri, S. Deep learning techniques for skin lesion analysis and melanoma cancer detection: a survey of state-of-the-art. en. *Artificial Intelligence Review* **54,** 811–841 (2021) (cited on pages 148, 193).
6. Aggarwal, U., Popescu, A. & Hudelot, C. *Active Learning for Imbalanced Datasets* en. in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* (IEEE, Snowmass Village, CO, USA, 2020), 1417–1426. (2023) (cited on page 147).
7. Agrawal, A., Chauhan, A., Shetty, M. K., P, G. M., Gupta, M. D. & Gupta, A. ECG-iCOVIDNet: Interpretable AI model to identify changes in the ECG signals of post-COVID subjects. en. *Computers in Biology and Medicine* **146.** (2024) (2022) (cited on pages 211, 215).
8. Albrecht, J. *Reduktion des Annotationsaufwands - Ein Vergleich von Active Learning Frameworks* Bachelorthesis (University of Augsburg, 2020) (cited on page 16).
9. Alday, E. A. P., Gu, A., Shah, A. J., Robichaux, C., Wong, A.-K. I., Liu, C., Liu, F., Rad, A. B., Elola, A., Seyedi, S., Li, Q., Sharma, A., Clifford, G. D. & Reyna, M. A. Classification of 12-lead ECGs: the PhysioNet/Computing in Cardiology Challenge 2020. *Physiological Measurement* **41,** 124003 (Dec. 2020) (cited on pages 161, 164, 182).
10. Ali, A.-R., Li, J. & O'Shea, S. J. Towards the automatic detection of skin lesion shape asymmetry, color variegation and diameter in dermoscopic images. en. *PLOS ONE* **15** (ed Haass, N. K.) (2020) (cited on page 201).
11. Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J. M., Confalonieri, R., Guidotti, R., Del Ser, J., Díaz-Rodríguez, N. & Herrera, F. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. en. *Information Fusion* **99,** 101805. (2024) (2023) (cited on page 42).
12. Almaraz-Damian, J.-A., Ponomaryov, V., Sadovnychiy, S. & Castillejos-Fernandez, H. Melanoma and Nevus Skin Lesion Classification Using Handcraft and Deep Learning Feature Fusion via Mutual Information Measures. *Entropy* **22,** 484 (Apr. 2020) (cited on page 201).
13. Alonso, O. Challenges with Label Quality for Supervised Learning. en. *Journal of Data and Information Quality* **6,** 1–3. (2023) (2015) (cited on pages 67, 101, 147).
14. Amann, J., Blasimme, A., Vayena, E., Frey, D. & Madai, V. I. Explainability for artificial intelligence in healthcare: a multidisciplinary perspective. en. *BMC Medical Informatics and Decision Making* **20,** 310 (2020) (cited on page 205).
15. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B. & Zimmermann, T. *Software Engineering for Machine Learning: A Case Study* in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (IEEE, Montreal, QC, Canada, 2019), 291–300. (2022) (cited on pages 65, 67, 71, 85, 87, 89).
16. Anaconda Inc. *Anaconda Software Distribution* 2016. https://anaconda.com (2023) (cited on page 138).

17. Angelaki, E., Marketou, M. E., Barmparis, G. D., Patrianakos, A., Vardas, P. E., Parthenakis, F. & Tsironis, G. P. Detection of abnormal left ventricular geometry in patients without cardiovascular disease through machine learning: An ECG-based approach. *The Journal of Clinical Hypertension* **23,** 935–945 (2021) (cited on page 211).

18. Angluin, D. Queries and concept learning. en. *Machine Learning* **2,** 319–342 (1988) (cited on page 31).

19. Annessi, G., Bono, R., Sampogna, F., Faraggiana, T. & Abeni, D. Sensitivity, specificity, and diagnostic accuracy of three dermoscopic algorithmic methods in the diagnosis of doubtful melanocytic lesions - The importance of light brown structureless areas in differentiating atypical melanocytic nevi from thin melanomas. **56,** 759–767 (2007) (cited on page 188).

20. Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., Torres, J., Van Hovell, H., Ionescu, A., Łuszczak, A., Świtakowski, M., Szafrański, M., Li, X., Ueshin, T., Mokhtar, M., Boncz, P., Ghodsi, A., Paranjpye, S., Senster, P., Xin, R. & Zaharia, M. Delta lake: high-performance ACID table storage over cloud object stores. en. *Proceedings of the VLDB Endowment* **13,** 3411–3424 (2020) (cited on page 90).

21. Arnold, M., Boston, J., Desmond, M., Duesterwald, E., Elder, B., Murthi, A., Navratil, J. & Reimer, D. *Towards Automating the AI Operations Lifecycle* arXiv:2003.12808 [cs]. 2020. http://arxiv.org/abs/2003.12808 (2022) (cited on pages 59, 71, 106).

22. Arpteg, A., Brinne, B., Crnkovic-Friis, L. & Bosch, J. *Software Engineering Challenges of Deep Learning* en. in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* arXiv:1810.12034 [cs] (2018), 50–59. (2022) (cited on pages 4, 85).

23. Arrieta, A. B., Díaz-Rodríguez, N., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R. & Herrera, F. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* **58,** 82–115 (2020) (cited on pages 43, 205).

24. Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q. V., Luss, R., Mojsilović, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varshney, K. R., Wei, D. & Zhang, Y. *One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques* arXiv:1909.03012 [cs, stat]. 2019. http://arxiv.org/abs/1909.03012 (cited on pages 43, 209).

25. Ashmore, R., Calinescu, R. & Paterson, C. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. en. *ACM Computing Surveys* **54,** 111:1–111:39. (2023) (2022) (cited on pages 59, 65, 72).

26. Atighehchian, P., Branchaud-Charron, F. & Lacoste, A. *Bayesian active learning for production, a systematic study and a reusable library* in (Vienna, Austria, 2020) (cited on pages 124, 126).

27. Atlas, L., Cohn, D., Ladner, R., El-Sharkawi, M. A., Marks, R. J., Aggoune, M. E. & Park, D. C. *Training connectionist networks with queries and selective sampling* in *Proceedings of the 2nd International Conference on Neural Information Processing Systems* (MIT Press, Cambridge, MA, USA, 1989), 566–573 (cited on page 32).

28. Azad, N. & Hyrynsalmi, S. DevOps critical success factors — A systematic literature review. en. *Information and Software Technology* **157,** 107150 (2023) (cited on page 52).

29. Barata, C., Celebi, M. E. & Marques, J. S. A Survey of Feature Extraction in Dermoscopy Image Analysis of Skin Cancer. en. *IEEE Journal of Biomedical and Health Informatics* **23,** 1096–1109 (2019) (cited on page 201).

30. Barata, R., Leite, M., Pacheco, R., Sampaio, M. O. P., Ascensão, J. T. & Bizarro, P. *Active learning for imbalanced data under cold start* in *Proceedings of the Second ACM International Conference on AI in Finance* (ACM, Virtual Event, 2021). (2023) (cited on page 62).

31. Bayer, M. in *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks* (eds Brown, A. & Wilson, G.) (aosabook.org, 2012) (cited on page 133).

32. Baylor, D., Breck, E., Cheng, H.-T., Fiedel, N., Foo, C. Y., Haque, Z., Haykal, S., Ispir, M., Jain, V., Koc, L., Koo, C. Y., Lew, L., Mewald, C., Modi, A. N., Polyzotis, N., Ramesh, S., Roy, S., Whang, S. E., Wicke, M., Wilkiewicz, J., Zhang, X. & Zinkevich, M. *TFX: A TensorFlow-Based Production-Scale Machine Learning Platform* en. in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, Halifax NS Canada, 2017), 1387–1395. (2023) (cited on pages 69, 80, 100, 106, 107).

33. Bell, A., Solano-Kamaiko, I., Nov, O. & Stoyanovich, J. *It's Just Not That Simple: An Empirical Study of the Accuracy-Explainability Trade-off in Machine Learning for Public Policy* en. in *2022 ACM Conference on Fairness, Accountability, and Transparency* (ACM, Seoul Republic of Korea, 2022), 248–266 (cited on page 7).

34. Beluch, W. H., Genewein, T., Nurnberger, A. & Kohler, J. M. *The Power of Ensembles for Active Learning in Image Classification* en. in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, Salt Lake City, UT, 2018), 9368–9377 (cited on pages 155, 172).

35. Benington, H. D. Production of Large Computer Programs. en. *IEEE Annals of the History of Computing* **5,** 350–361 (1983) (cited on page 49).

36. Bergmair, J. *Model Explanations As Quality Gate in Machine Learning Pipelines* MA thesis (University of Augsburg, 2022) (cited on page 17).

37. Bernhardt, M., Castro, D. C., Tanno, R., Schwaighofer, A., Tezcan, K. C., Monteiro, M., Bannur, S., Lungren, M. P., Nori, A., Glocker, B., Alvarez-Valle, J. & Oktay, O. Active label cleaning for improved dataset quality under resource constraints. en. *Nature Communications* **13,** 1161. (2023) (2022) (cited on pages 67, 101, 147).

38. Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M. F. & Eckersley, P. *Explainable machine learning in deployment* in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (ACM, Barcelona Spain, 2020), 648–657 (cited on pages 205–207).

39. Bienefeld, N., Boss, J. M., Lüthy, R., Brodbeck, D., Azzati, J., Blaser, M., Willms, J. & Keller, E. Solving the explainable AI conundrum by bridging clinicians' needs and developers' goals. *npj Digital Medicine* **6** (2023) (cited on page 218).

40. Bishop, C. M. *Pattern recognition and machine learning* en (Springer, New York, 2006) (cited on page 26).

41. Bizopoulos, P. & Koutsouris, D. Deep Learning in Cardiology. en. *IEEE Reviews in Biomedical Engineering* **12,** 168–193 (2019) (cited on page 161).

42. Booch, G. & Brown, A. W. Collaborative Development Environments. *Doctor Dobbs Journal* (2003) (cited on page 108).

43. Brangbour, E., Bruneau, P., Tamisier, T. & Marchand-Maillet, S. en. in *Cooperative Design, Visualization, and Engineering* (ed Luo, Y.) Series Title: Lecture Notes in Computer Science, 192–201 (Springer International Publishing, Cham, 2020). (2023) (cited on page 63).

44. Breck, E., Cai, S., Nielsen, E., Salib, M. & Sculley, D. What's your ML Test Score? A rubric for ML production systems (2017) (cited on pages 75, 100, 103).

45. Breck, E., Cai, S., Nielsen, E., Salib, M. & Sculley, D. *The ML test score: A rubric for ML production readiness and technical debt reduction* in *2017 IEEE International Conference on Big Data (Big Data)* (IEEE, Boston, MA, 2017), 1123–1132 (cited on pages 100, 102, 104, 105, 109).

46. Breck, E., Polyzotis, N., Roy, S., Whang, S. E. & Zinkevich, M. *Data Validation for Machine Learning* en. in *Proceedings of the 2nd SysML Conference* (Palo Alto, CA, USA, 2019) (cited on pages 66, 70, 100).

47. Brinker, T. J., Hekler, A., Utikal, J. S., Grabe, N., Schadendorf, D., Klode, J., Berking, C., Steeb, T., Enk, A. H. & Von Kalle, C. Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review. en. *Journal of Medical Internet Research* **20,** e11936 (2018) (cited on page 148).

48. Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., Khlaaf, H., Yang, J., Toner, H., Fong, R., Maharaj, T., Koh, P. W., Hooker, S., Leung, J., Trask, A., Bluemke, E., Lebensold, J., O'Keefe, C., Koren, M., Ryffel, T., Rubinovitz, J. B., Besiroglu, T., Carugati, F., Clark, J., Eckersley, P., de Haas, S., Johnson, M., Laurie, B., Ingerman, A., Krawczuk, I., Askell, A., Cammarota, R., Lohn, A., Krueger, D., Stix, C., Henderson, P., Graham, L., Prunkl, C., Martin, B., Seger, E., Zilberman, N., hÉigeartaigh, S. Ó., Kroeger, F., Sastry, G., Kagan, R., Weller, A., Tse, B., Barnes, E., Dafoe, A., Scharre, P., Herbert-Voss, A., Rasser, M., Sodhani, S., Flynn, C., Gilbert, T. K., Dyer, L., Khan, S., Bengio, Y. & Anderljung, M. *Toward Trustworthy AI Development: Mechanisms for Supporting Verifiable Claims* arXiv:2004.07213 [cs]. 2020. http://arxiv.org/abs/2004.07213 (cited on page 3).

49. Budd, S., Robinson, E. C. & Kainz, B. A Survey on Active Learning and Human-in-the-Loop Deep Learning for Medical Image Analysis. en. *Medical Image Analysis* **71.** arXiv:1910.02923 [cs, eess], 102062. (2023) (2021) (cited on page 81).

50. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B. & Varoquaux, G. *API design for machine learning software: experiences from the scikit-learn project* in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013), 108–122 (cited on page 141).

51. Carvalho, D. V., Pereira, E. M. & Cardoso, J. S. Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics* **8,** 832 (2019) (cited on page 43).

52. Cassidy, B., Kendrick, C., Brodzicki, A., Jaworek-Korjakowska, J. & Yap, M. H. Analysis of the ISIC image datasets: Usage, benchmarks and recommendations. en. *Medical Image Analysis* **75,** 102305. (2024) (2022) (cited on page 150).

53. Caveness, E., G. C., P. S., Peng, Z., Polyzotis, N., Roy, S. & Zinkevich, M. *TensorFlow Data Validation: Data Analysis and Validation in Continuous ML Pipelines* en. in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (ACM, Portland OR USA, 2020), 2793–2796. (2023) (cited on pages 70, 79).

54. Celebi, M. E., Barata, C., Halpern, A., Tschandl, P., Combalia, M. & Liu, Y. Guest Editorial Skin Image Analysis in the Age of Deep Learning. en. *IEEE Journal of Biomedical and Health Informatics* **27,** 143–144 (2023) (cited on page 148).

55. Chacon, S. & Straub, B. *Pro Git* 2nd ed. (Apress, 2014) (cited on pages 86, 90, 93, 137).

56. Charniak, E. *Introduction to Deep Learning* (The MIT Press, Cambridge, MA, 2018) (cited on pages 26, 27).

57. Chen, C., Zhao, H. Y., Zheng, S. H., Ramachandra, R. A., He, X., Zhang, Y. H. & Sudarshan, V. K. Interpretable hybrid model for an automated patient-wise categorization of hypertensive and normotensive electrocardiogram signals. en. *Computer Methods and Programs in Biomedicine Update* **3** (2023) (cited on page 210).

58. Chen, L., Bai, Y., Huang, S., Lu, Y., Wen, B., Yuille, A. L. & Zhou, Z. *Making Your First Choice: To Address Cold Start Problem in Vision Active Learning* en. arXiv:2210.02442 [cs]. 2022. http://arxiv.org/abs/2210.02442 (2023) (cited on page 63).

59. Chew, R., Wenger, M., Kery, C., Nance, J., Richards, K., Hadley, E. & Baumgartner, P. SMART: An Open Source Data Labeling Platform for Supervised Learning. *Journal of Machine Learning Research* **20,** 1–5. (2020) (2019) (cited on pages 124, 126).

60. Chromik, M. & Schuessler, M. A Taxonomy for Human Subject Evaluation of Black-Box Explanations in XAI. en (2020) (cited on page 45).

61. Cohn, D., Atlas, L. & Ladner, R. Improving generalization with active learning. en. *Machine Learning* **15,** 201–221 (1994) (cited on page 32).

62. Danka, T. & Horvath, P. modAL: A modular active learning framework for Python. *arXiv:1805.00979 [cs, stat].* arXiv: 1805.00979. (2020) (2018) (cited on pages 124, 126).

63. Debois, P. *Agile Infrastructure and Operations: How Infra-gile are You?* en. in *Agile 2008 Conference* (IEEE, Toronto, ON, Canada, 2008), 202–207 (cited on page 52).

64. Desmond, M., Duesterwald, E., Brimijoin, K., Brachman, M. & Pan, Q. Semi-Automated Data Labeling. en. *Journal of Machine Learning Research,* 156–169 (2021) (cited on page 67).

65. Dey, S., Chakraborty, P., Kwon, B. C., Dhurandhar, A., Ghalwash, M., Suarez Saiz, F. J., Ng, K., Sow, D., Varshney, K. R. & Meyer, P. Human-centered explainability for life sciences, healthcare, and medical informatics. *Patterns* **3,** 100493 (2022) (cited on pages 42, 205).

66. Dhanorkar, S., Wolf, C. T., Qian, K., Xu, A., Popa, L. & Li, Y. *Who needs to know what, when?: Broadening the Explainable AI (XAI) Design Space by Looking at Explanations Across the AI Lifecycle* in *Designing Interactive Systems Conference 2021* (ACM, Virtual Event USA, 2021), 1591–1602 (cited on page 203).

67. Diaz-de-Arcaya, J., Torre-Bastida, A. I., Zárate, G., Miñón, R. & Almeida, A. A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey. *ACM Computing Surveys.* (2023) (2023) (cited on pages 4, 57, 85).

68. Dijk, O., oegesam, Bell, R., Lily, Simon-Free, Serna, B., rajgupt, yanhong-zhao-ef, Gädke, A., Todor, A., Evgeniy, Hugo, Haizad, M., Okumus, T. & woochan-jang. *ExplainerDashboard* 2023. https://doi.org/10.5281/zenodo.7633294 (cited on page 210).

69. Doshi-Velez, F. & Kim, B. *Towards A Rigorous Science of Interpretable Machine Learning* arXiv:1702.08608 [cs, stat]. 2017 (cited on pages 45, 216).

70. Dozat, T. *Incorporating Nesterov Momentum into Adam* en. Tech. rep. (2015) (cited on pages 28, 170).

71. Driessen, V. *A successful Git branching model* 2010. https://nvie.com/posts/a-successful-git-branching-model/ (2023) (cited on page 86).

72. Druck, G., Settles, B. & McCallum, A. *Active learning by labeling features* en. in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing Volume 1 - EMNLP '09* **1** (Association for Computational Linguistics, Singapore, 2009), 81 (cited on page 31).

73. Ebert, S., Fritz, M. & Schiele, B. *RALF: A reinforced active learning formulation for object class recognition* en. in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Providence, RI, 2012), 3626–3633 (cited on pages 37, 155).

74. Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. Least angle regression. en. *The Annals of Statistics* **32,** 407–499 (2004) (cited on page 190).

75. Elia, M. *Evaluating Explainability in the Context of Active Learning* MA thesis (University of Augsburg, 2021) (cited on page 16).

76. Elia, M., Peter, T., Stieler, F., Bauer, B., Nagl, S., Ebigbo, A. & Grünherz, V. *Precision medicine for achalasia diagnosis: a multi-modal and interdisciplinary approach for training data generation [Abstract]* in *IEEE - ISBI 2023: International Symposium on Biomedical Imaging, Cartagena de Indias, Colombia, April 18-21, 2023* (2023) (cited on page 22).

77. Elia, M., Stieler, F., Ripke, F., Nann, M., Dopfer, S. & Bauer, B. *Towards Certifiable AI in Medicine: Illustrated for Multi-label ECG Classification Performance Metrics* in *Proceedings of the IEEE International Conference on Evolving and Adaptive Intelligent Systems 2024* (Madrid, Spain, 2024) (cited on page 22).

78. Ereth, J. *DataOps – Towards a Definition* en. in *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen"* **2191** (Mannheim, 2018), 104–112 (cited on page 52).

79. Ertekin, S., Huang, J., Bottou, L. & Giles, L. *Learning on the border: active learning in imbalanced data classification* en. in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (ACM, Lisbon Portugal, 2007), 127–136 (cited on page 147).

80. Ertekin, S., Huang, J. & Giles, C. L. *Active learning for class imbalance problem* en. in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (ACM, Amsterdam The Netherlands, 2007), 823–824 (cited on page 147).

81. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M. & Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. en. *Nature* **542,** 115–118 (2017) (cited on page 148).

82. European Commission. *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS* 2024. `https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:52021PC0206` (2024) (cited on pages 39, 206).

83. Faraj, S. & Sproull, L. Coordinating Expertise in Software Development Teams. *Management Science* **46,** 1554–1568 (2000) (cited on page 108).

84. Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. From Data Mining to Knowledge Discovery in Databases. en. *AI Magazine* **17,** 37–54 (1996) (cited on pages 49, 50, 66).

85. Ferreira, J. J. & Monteiro, M. S. *What Are People Doing About XAI User Experience? A Survey on AI Explainability Research and Practice* in *Design, User Experience, and Usability. Design for Contemporary Interactive Environments* (eds Marcus, A. & Rosenzweig, E.) (Springer International Publishing, Cham, 2020), 56–73 (cited on pages 45, 207).

86. Figel, L. *Bias Detection using Interpretable Machine Learning Methods* Bachelorthesis (University of Augsburg, 2021) (cited on page 16).

87. Fischer, L., Ehrlinger, L., Geist, V., Ramler, R., Sobiezky, F., Zellinger, W., Brunner, D., Kumar, M. & Moser, B. AI System Engineering—Key Challenges and Lessons Learned. *Machine Learning and Knowledge Extraction* **3,** 56–83. (2022) (2020) (cited on pages 67, 85).

88. Floridi, L. in *Ethics, Governance, and Policies in Artificial Intelligence* (ed Floridi, L.) 41–45 (Springer International Publishing, Cham, 2021) (cited on page 42).

89. Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., Schafer, B., Valcke, P. & Vayena, E. AI4People—An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations. en. *Minds and Machines* **28,** 689–707 (2018) (cited on page 39).

90. Fong, R. & Vedaldi, A. Interpretable Explanations of Black Boxes by Meaningful Perturbation. *2017 IEEE International Conference on Computer Vision (ICCV),* 3449–3457 (2017) (cited on page 190).

91. Forsberg, K. & Mooz, H. The Relationship of System Engineering to the Project Cycle. en. *INCOSE International Symposium* **1,** 57–65 (1991) (cited on page 49).

92. Fredriksson, T., Bosch, J. & Olsson, H. *Machine Learning Models for Automatic Labeling: A Systematic Literature Review:* en. in *Proceedings of the 15th International Conference on Software Technologies* (SCITEPRESS - Science and Technology Publications, Lieusaint - Paris, France, 2020), 552–561. (2023) (cited on page 67).

93. Fredriksson, T., Mattos, D. I., Bosch, J. & Olsson, H. H. en. in *Product-Focused Software Process Improvement* (eds Morisio, M., Torchiano, M. & Jedlitschka, A.) Series Title: Lecture Notes in Computer Science, 202–216 (Springer International Publishing, Cham, 2020). (2022) (cited on pages 4, 81).

94. Freeman, B., Hammel, N., Phene, S., Huang, A., Ackermann, R., Kanzheleva, O., Hutson, M., Taggart, C., Duong, Q. & Sayres, R. Iterative Quality Control Strategies for Expert Medical Image Labeling. en. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* **9,** 60–71 (2021) (cited on page 147).

95. Friedman, R. J., Rigel, D. S. & Kopf, A. W. Early Detection of Malignant Melanoma: The Role of Physician Examination and Self-Examination of the Skin. *CA: A Cancer Journal for Clinicians* **35,** 130–151 (1985) (cited on page 188).

96. Fu, Y., Zhu, X. & Li, B. A survey on instance selection for active learning. en. *Knowledge and Information Systems* **35,** 249–283 (2013) (cited on page 34).

97. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A. A survey on concept drift adaptation. en. *ACM Computing Surveys* **46,** 1–37. (2023) (2014) (cited on page 77).

98. Gareau, D. S., Browning, J., Correa Da Rosa, J., Suarez-Farinas, M., Lish, S., Zong, A. M., Firester, B., Vrattos, C., Renert-Yuval, Y., Gamboa, M., Vallone, M. G., Barragán-Estudillo, Z. F., Tamez-Peña, A. L., Montoya, J., Jesús-Silva, M. A., Carrera, C., Malvehy, J., Puig, S., Marghoob, A., Carucci, J. A. & Krueger, J. G. Deep learning-level melanoma detection by interpretable machine learning and imaging biomarker cues. en. *Journal of Biomedical Optics* **25.** (2021) (2020) (cited on pages 186, 188).

99. Ge, Y., Liu, S., Fu, Z., Tan, J., Li, Z., Xu, S., Li, Y., Xian, Y. & Zhang, Y. A Survey on Trustworthy Recommender Systems. en. *ACM Transactions on Recommender Systems,* 3652891 (2024) (cited on page 41).

100. Ghai, B., Liao, Q. V., Zhang, Y., Bellamy, R. & Mueller, K. Explainable Active Learning (XAL): Toward AI Explanations as Interfaces for Machine Teachers. en. *Proceedings of the ACM on Human-Computer Interaction* **4,** 1–28 (2021) (cited on pages 31, 157).

101. Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M. & Kagal, L. *Explaining Explanations: An Overview of Interpretability of Machine Learning* en. in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)* (IEEE, Turin, Italy, 2018), 80–89. (2024) (cited on page 203).

102. Giray, G. A software engineering perspective on engineering machine learning systems: State of the art and challenges. en. *Journal of Systems and Software* **180.** (2022) (2021) (cited on pages 86, 90, 110).

103. GitHub. *Introduction to dev containers* 2023. https://docs.github.com/en/codespaces/setting-up-your-project-for-codespaces/adding-a-dev-container-configuration/introduction-to-dev-containers (2023) (cited on page 75).

104. GitLab. *CI/CD analytics* 2023. https://docs.gitlab.com/ee/user/analytics/ci_cd_analytics.html (2023) (cited on page 117).

105. Gleicher, M. A Framework for Considering Comprehensibility in Modeling. en. *Big Data* **4,** 75–88. (2024) (2016) (cited on pages 45, 204, 205).

106. Gmeiner, J., Ramler, R. & Haslinger, J. *Automated testing in the continuous delivery pipeline: A case study of an online company* en. in *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (IEEE, Graz, Austria, 2015), 1–6. (2023) (cited on page 75).

107. Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K. & Stanley, H. E. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. en. *Circulation* **101** (2000) (cited on pages 161, 162).

108. Gomes, I. V., Morgado, P., Gomes, T. & Moreira, R. M. L. M. *An overview on the Static Code Analysis approach in Software Development* en. 2009 (cited on page 105).

109. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016) (cited on pages 28–30).

110. Google. *MLOps: Continuous delivery and automation pipelines in machine learning* 2023. https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning (2023) (cited on pages 111, 112).

111. Google. *Responsible AI practices* 2023. https://ai.google/responsibility/responsible-ai-practices/ (2023) (cited on pages 100, 102, 104, 107, 109).

112. Gorodkin, J. Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry* **28,** 367–374 (2004) (cited on page 172).

113. Goyal, M., Knackstedt, T., Yan, S. & Hassanpour, S. Artificial intelligence-based image classification methods for diagnosis of skin cancer: Challenges and opportunities. en. *Computers in Biology and Medicine* **127,** 104065 (2020) (cited on page 148).

114. Greenfeld, A. R. *cookiecutter* 2013. https://cookiecutter.readthedocs.io (2023) (cited on page 129).

115. Grünherz, V., Ebigbo, A., Elia, M., Brunner, A., Krafft, T., Pöller, L., Schneider, P., Stieler, F., Bauer, B., Muzalyova, A., Messmann, H. & Nagl, S. Automatic three-dimensional reconstruction of the oesophagus in achalasia patients undergoing POEM: an innovative approach for evaluating treatment outcomes. en. *BMJ Open Gastroenterology* **11,** e001396. (2024) (2024) (cited on page 22).

116. Haakman, M., Cruz, L., Huijgens, H. & van Deursen, A. AI lifecycle models need to be revised: An exploratory study in Fintech. en. *Empirical Software Engineering* **26,** 95. (2023) (2021) (cited on pages 4, 57, 69).

117. Habibullah, K. M., Gay, G. & Horkoff, J. Non-functional requirements for machine learning: understanding current use and challenges among practitioners. en. *Requirements Engineering* **28,** 283–316. (2023) (2023) (cited on pages 5, 74).

118. Hagendorff, T. The Ethics of AI Ethics: An Evaluation of Guidelines. *Minds and Machines* **30,** 99–120 (2020) (cited on page 42).

119. Haggenmüller, S., Maron, R. C., Hekler, A., Utikal, J. S., Barata, C., Barnhill, R. L., Beltraminelli, H., Berking, C., Betz-Stablein, B., Blum, A., Braun, S. A., Carr, R., Combalia, M., Fernandez-Figueras, M.-T., Ferrara, G., Fraitag, S., French, L. E., Gellrich, F. F., Ghoreschi, K., Goebeler, M., Guitera, P., Haenssle, H. A., Haferkamp, S., Heinzerling, L., Heppt, M. V., Hilke, F. J., Hobelsberger, S., Krahl, D., Kutzner, H., Lallas, A., Liopyris, K., Llamas-Velasco, M., Malvehy, J., Meier, F., Müller, C. S., Navarini, A. A., Navarrete-Dechent, C., Perasole, A., Poch, G., Podlipnik, S., Requena, L., Rotemberg, V. M., Saggini, A., Sangueza, O. P., Santonja, C., Schadendorf, D., Schilling, B., Schlaak, M., Schlager, J. G., Sergon, M., Sondermann, W., Soyer, H. P., Starz, H., Stolz, W., Vale, E., Weyers, W., Zink, A., Krieghoff-Henning, E., Kather, J. N., Von Kalle, C., Lipka, D. B., Fröhling, S., Hauschild, A., Kittler, H. & Brinker, T. J. Skin cancer classification via convolutional neural networks: systematic review of studies involving human experts. en. *European Journal of Cancer* **156,** 202–216 (2021) (cited on page 148).

120. Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P. & Ng, A. Y. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine* **25,** 65–69 (2019) (cited on page 161).

121. Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. E. Array programming with NumPy. en. *Nature* **585,** 357–362 (2020) (cited on page 141).

122. Hauser, K., Kurz, A., Haggenmüller, S., Maron, R. C., Von Kalle, C., Utikal, J. S., Meier, F., Hobelsberger, S., Gellrich, F. F., Sergon, M., Hauschild, A., French, L. E., Heinzerling, L., Schlager, J. G., Ghoreschi, K., Schlaak, M., Hilke, F. J., Poch, G., Kutzner, H., Berking, C., Heppt, M. V., Erdmann, M., Haferkamp, S., Schadendorf, D., Sondermann, W., Goebeler, M., Schilling, B., Kather, J. N., Fröhling, S., Lipka, D. B., Hekler, A., Krieghoff-Henning, E. & Brinker, T. J. Explainable artificial intelligence in skin cancer recognition: A systematic review. *European Journal of Cancer* **167,** 54–69 (2022) (cited on page 188).

123. He, X., Zhao, K. & Chu, X. AutoML: A Survey of the State-of-the-Art. en. *Knowledge-Based Systems* **212,** 106622. (2023) (2021) (cited on page 63).

124. Henning, J. S., Dusza, S. W., Wang, S. Q., Marghoob, A. A., Rabinovitz, H. S., Polsky, D. & Kopf, A. W. The CASH (color, architecture, symmetry, and homogeneity) algorithm for dermoscopy. *Journal of the American Academy of Dermatology* **56,** 45–52 (2007) (cited on page 188).

125. Hesenius, M., Schwenzfeier, N., Meyer, O., Koop, W. & Gruhn, V. *Towards a Software Engineering Process for Developing Data-Driven Applications* en. in *2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)* (IEEE, Montreal, QC, Canada, 2019), 35–41. (2024) (cited on page 51).

126. Hickman, E. & Petrin, M. Trustworthy AI and Corporate Governance: The EU's Ethics Guidelines for Trustworthy Artificial Intelligence from a Company Law Perspective. en. *European Business Organization Law Review* **22,** 593–625 (2021) (cited on page 38).

127. High-Level Expert Group on AI. *Ethics Guidelines For Trustworthy AI* (European Commission, 2019) (cited on pages iii, v, 3–7, 38–40, 42).

128. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **9,** 1735–1780 (1997) (cited on page 29).

129. Hochschule für Philosophie München / Philosophische Fakultät. *Center for Responsible AI Technologies* 2022. https://www.center-responsible-ai.de/ (2023) (cited on page 15).

130. Hong, S. R., Hullman, J. & Bertini, E. Human Factors in Model Interpretability: Industry Practices, Challenges, and Needs. *Proceedings of the ACM on Human-Computer Interaction* **4.** arXiv:2004.11440 [cs] (2020) (cited on pages 204–208).

131. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* arXiv:1704.04861 [cs]. 2017. http://arxiv.org/abs/1704.04861 (cited on page 193).

132. Hsu, W.-N. & Lin, H.-T. Active Learning by Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* **29** (2015) (cited on pages 35, 124, 126).

133. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. *Densely Connected Convolutional Networks* in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Honolulu, HI, 2017), 2261–2269 (cited on pages 154, 182).

134. Huang, K.-H. *DeepAL: Deep Active Learning in Python* arXiv:2111.15258 [cs]. 2021. http://arxiv.org/abs/2111.15258 (cited on pages 124, 126).

135. Hughes, J. W., Olgin, J. E., Avram, R., Abreau, S. A., Sittler, T., Radia, K., Hsia, H., Walters, T., Lee, B., Gonzalez, J. E. & Tison, G. H. Performance of a Convolutional Neural Network and Explainability Technique for 12-Lead Electrocardiogram Interpretation. *JAMA Cardiology* **6,** 1285–1295 (2021) (cited on pages 210, 215).

136. Hummer, W., Muthusamy, V., Rausch, T., Dube, P., El Maghraoui, K., Murthi, A. & Oum, P. *ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI* in *2019 IEEE International Conference on Cloud Engineering (IC2E)* (IEEE, Prague, Czech Republic, 2019), 113–120 (cited on pages 69, 104, 107).

137. Hutter, F., Kotthoff, L. & Vanschoren, J. *Automated Machine Learning: Methods, Systems, Challenges* en (Springer International Publishing, Cham, 2019) (cited on page 103).

138. IBM Corporation. *Analytics Solutions Unified Method: Implementations with Agile principles.* 2016 (cited on page 51).

139. Ibrahim, L., Mesinovic, M., Yang, K.-W. & Eid, M. A. Explainable Prediction of Acute Myocardial Infarction Using Machine Learning and Shapley Values. en. *IEEE Access* **8,** 210410–210417 (2020) (cited on page 211).

140. Idowu, S., Strüber, D. & Berger, T. *Asset Management in Machine Learning: A Survey* en. arXiv:2102.06919 [cs]. 2021. http://arxiv.org/abs/2102.06919 (cited on pages 65, 71).

141. IEEE Standards Board. *IEEE Standard for Developing Software Life Cycle Processes* en. Tech. rep. ISBN: 9780738104140 (IEEE, 1995) (cited on page 58).

142. Ienco, D., Bifet, A., Žliobaitė, I. & Pfahringer, B. en. in *Discovery Science* (eds Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Fürnkranz, J., Hüllermeier, E. & Higuchi, T.) 79–93 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013) (cited on page 37).

143. Ilyas, I. F. & Chu, X. *Data Cleaning* (Association for Computing Machinery, New York, NY, USA, 2019) (cited on page 66).

144. Isdahl, R. & Gundersen, O. E. *Out-of-the-Box Reproducibility: A Survey of Machine Learning Platforms* in *2019 15th International Conference on eScience (eScience)* (IEEE, San Diego, CA, USA, 2019), 86–95 (cited on page 6).

145. ISO/IEC TR 24028:2020. *Information Technology–Artificial Intelligence–Overview of Trustworthiness in Artificial Intelligence* 2020 (cited on pages 38, 41).

146. ISO/IEC TR 29119-11:2020. *Software and systems engineering - Software testing - Part 11: Guidelines on the testing of AI-based systems* 2020 (cited on page 41).

147. Iterative. *DVC: Data Version Control - Git for Data & Models* 2020. DOI:10.5281/zenodo.012345 (cited on pages 87, 90, 117, 137, 139, 141, 144).

148. Iterative. *Data Registry* 2023. https://dvc.org/doc/use-cases/data-registry#data-registry (2023) (cited on page 139).

149. Iterative. *DVC Command Reference* 2023. https://dvc.org/doc/command-reference (2023) (cited on page 139).

150. Iterative. *Internal Directories and Files* 2024. https://dvc.org/doc/user-guide/project-structure/internal-files (2024) (cited on pages 152, 165).

151. Ivaturi, P., Gadaleta, M., Pandey, A. C., Pazzani, M., Steinhubl, S. R. & Quer, G. A Comprehensive Explanation Framework for Biomedical Time Series Classification. en. *IEEE Journal of Biomedical and Health Informatics* **25,** 2398–2408 (2021) (cited on pages 211, 212).

152. Jaccard, P. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Société vaudoise des sciences naturelles* **37,** 547–579 (1901) (cited on page 172).

153. James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning with Applications in R* (eds G., C., S., F. & I., O.) (Springer New York, NY, 2013) (cited on pages 4, 24, 25).

154. Jekova, I., Christov, I. & Krasteva, V. Atrioventricular Synchronization for Detection of Atrial Fibrillation and Flutter in One to Twelve ECG Leads Using a Dense Neural Network Classifier. en. *Sensors* **22** (2022) (cited on page 211).

155. Jeong, H. K., Park, C., Henao, R. & Kheterpal, M. Deep Learning in Dermatology: A Systematic Review of Current Approaches, Outcomes, and Limitations. en. *JID Innovations* **3,** 100150 (2023) (cited on page 154).

156. Jiang, S., Li, H. & Jin, Z. A Visually Interpretable Deep Learning Framework for Histopathological Image-Based Skin Cancer Diagnosis. *IEEE Journal of Biomedical and Health Informatics* **25,** 1483–1494 (2021) (cited on page 186).

157. John, M. M., Olsson, H. H. & Bosch, J. *Towards MLOps: A Framework and Maturity Model* en. in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (IEEE, Palermo, Italy, 2021), 334–341. (2023) (cited on pages 53, 111, 112).

158. Johs, A. J., Agosto, D. E. & Weber, R. O. *Qualitative Investigation in Explainable Artificial Intelligence: A Bit More Insight from Social Science* in *Proceedings for the Explainable Agency in AI Workshop at the 35th AAAI Conference on Artificial Intelligence* arXiv:2011.07130 [cs] (2021), 163–173 (cited on page 218).

159. Kale, D. & Liu, Y. *Accelerating Active Learning with Transfer Learning* en. in *2013 IEEE 13th International Conference on Data Mining* (IEEE, Dallas, TX, USA, 2013), 1085–1090. (2023) (cited on page 64).

160. Karamitsos, I., Albarhami, S. & Apostolopoulos, C. Applying DevOps Practices of Continuous Automation for Machine Learning. en. *Information* **11,** 363. (2023) (2020) (cited on pages 53, 87).

161. Karlaš, B., Interlandi, M., Renggli, C., Wu, W., Zhang, C., Mukunthu Iyappan Babu, D., Edwards, J., Lauren, C., Xu, A. & Weimer, M. *Building Continuous Integration Services for Machine Learning* en. in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (ACM, Virtual Event CA USA, 2020), 2407–2415. (2022) (cited on pages 72, 75, 87).

162. Kaur, D., Uslu, S. & Durresi, A. *Requirements for Trustworthy Artificial Intelligence – A Review* in *Advances in Networked-Based Information Systems* (eds Barolli, L., Li, K. F., Enokido, T. & Takizawa, M.) (Springer International Publishing, Cham, 2021), 105–115 (cited on page 38).

163. Kaur, D., Uslu, S., Rittichier, K. J. & Durresi, A. Trustworthy Artificial Intelligence: A Review. en. *ACM Computing Surveys* **55,** 1–38 (2023) (cited on pages 3, 38).

164. Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H. & Wortman Vaughan, J. *Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning* in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (ACM, Honolulu HI USA, 2020) (cited on page 206).

165. Keycloak Authors. *Keycloak - Open Source Identity and Access Management* 2023. https://www.keycloak.org/ (2023) (cited on pages 130, 166).

166. Khater, T., Ansari, S., Mahmoud, S., Hussain, A. & Tawfik, H. Skin cancer classification using explainable artificial intelligence on pre-extracted image features. *Intelligent Systems with Applications* **20** (2023) (cited on page 186).

167. Khomh, F., Adams, B., Cheng, J., Fokaefs, M. & Antoniol, G. Software Engineering for Machine-Learning Applications: The Road Ahead. *IEEE Software* **35,** 81–84 (2018) (cited on page 101).

168. Kienzler, R. *The lightweight IBM Cloud Garage Method for data science* 2020. https://developer.ibm.com/articles/the-lightweight-ibm-cloud-garage-method-for-data-science/ (cited on page 51).

169. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F. & Sayres, R. *Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)* arXiv:1711.11279 [stat]. 2018 (cited on page 186).

170. King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., Sparkes, A., Whelan, K. E. & Clare, A. The Automation of Science. en. *Science* **324,** 85–89. (2024) (2009) (cited on page 32).

171. King, R. D., Whelan, K. E., Jones, F. M., Reiser, P. G. K., Bryant, C. H., Muggleton, S. H., Kell, D. B. & Oliver, S. G. Functional genomic hypothesis generation and experimentation by a robot scientist. en. *Nature* **427,** 247–252 (2004) (cited on page 32).

172. Kingma, D. P. & Ba, J. L. *Adam: A Method for Stochastic Optimization* in. arXiv:1412.6980 [cs] (San Diego, 2015) (cited on pages 27, 28, 154).

173. Kitzinger, J. Qualitative research. Introducing focus groups. *BMJ* **311.** Publisher: BMJ, 299–302 (1995) (cited on pages 218, 225).

174. Klaise, J., Looveren, A. V., Vacanti, G. & Coca, A. Alibi Explain: Algorithms for Explaining Machine Learning Models. *Journal of Machine Learning Research* **22** (2021) (cited on page 209).

175. Kreuzberger, D., Kühl, N. & Hirschl, S. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. en. *IEEE Access* **11,** 31866–31879. (2023) (2023) (cited on pages 6, 53, 66, 69, 76–78, 82, 86, 87, 110, 123).

176. Krippendorff, K. Bivariate Agreement Coefficients for Reliability of Data. *Sociological Methodology* **2** (eds Borgatta, E. F. & Bohrnstedt, G. W.) 139–150 (1970) (cited on page 180).

177. Krippendorff, K. *Computing Krippendorff's Alpha-Reliability* 2011 (cited on page 181).

178. Krueger, R. A. & Casey, M. A. *Focus Groups: A Practical Guide for Applied Research* (SAGE Publications, 2014) (cited on page 218).

179. Kumar, P. & Gupta, A. Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey. en. *Journal of Computer Science and Technology* **35,** 913–945 (2020) (cited on pages 34, 36).

180. Kutzias, D., Dukino, C., Kötter, F. & Kett, H. *Comparative Analysis of Process Models for Data Science Projects:* en. in *Proceedings of the 15th International Conference on Agents and Artificial Intelligence* (SCITEPRESS - Science and Technology Publications, Lisbon, Portugal, 2023), 1052–1062. (2024) (cited on pages 49, 51).

181. Laato, S., Tiainen, M., Najmul Islam, A. & Mäntymäki, M. How to explain AI systems to end users: a systematic literature review and research agenda. en. *Internet Research* **32** (2022) (cited on pages 7, 206–209).

182. Lang, K. J. & Baum, E. B. *Query learning can work poorly when a human oracle is used* in *Proceedings of the IEEE International Joint Conference on Neural Networks* (IEEE Press, 1992), 335–340 (cited on page 32).

183. Lang, N. *Evaluating Explainable AI Algorithms Using Scenario-Based Focus Groups* MA thesis (University of Augsburg, 2023) (cited on page 17).

184. Langer, M., Oster, D., Speith, T., Hermanns, H., Kästner, L., Schmidt, E., Sesing, A. & Baum, K. What Do We Want From Explainable Artificial Intelligence (XAI)? – A Stakeholder Perspective on XAI and a Conceptual Model Guiding Interdisciplinary XAI Research. *Artificial Intelligence* **296.** arXiv:2102.07817 [cs] (2021) (cited on pages 45, 207).

185. Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W. & Müller, K.-R. Unmasking Clever Hans predictors and assessing what machines really learn. en. *Nature Communications* **10,** 1096 (2019) (cited on page 6).

186. Lazzeri, F. *How do teams work together on an automated machine learning project?* 2019. https://azure.microsoft.com/en-us/blog/how-do-teams-work-together-on-an-automated-machine-learning-project/ (cited on pages 101, 102, 104).

187. Lenarduzzi, V., Lomio, F., Moreschini, S., Taibi, D. & Tamburri, D. A. en. in *Software Quality: Future Perspectives on Software Engineering Quality* (eds Winkler, D., Biffl, S., Mendez, D., Wimmer, M. & Bergsmann, J.) Series Title: Lecture Notes in Business Information Processing, 43–53 (Springer International Publishing, Cham, 2021). (2023) (cited on pages 75, 82, 92, 93, 102).

188. Lewis, C. H. *Using the "Thinking Aloud" Method In Cognitive Interface Design* tech. rep. RC-9265 (IBM, Yorkton Heights, NY, USA, 1982) (cited on page 216).

189. Lewis, D. D. & Catlett, J. in *Machine Learning Proceedings 1994* (eds Cohen, W. W. & Hirsh, H.) 148–156 (Morgan Kaufmann, San Francisco (CA), 1994) (cited on pages 155, 173).

190. Lewis, D. D. & Gale, W. A. *A sequential algorithm for training text classifiers* in *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Springer-Verlag, London, 1994), 3–12 (cited on pages 33, 35).

191. Li, B., Qi, P., Liu, B., Di, S., Liu, J., Pei, J., Yi, J. & Zhou, B. Trustworthy AI: From Principles to Practices. en. *ACM Computing Surveys* **55,** 1–46 (2023) (cited on pages 3, 41).

192. Li, Z., Koban, K. C., Schenck, T. L., Giunta, R. E., Li, Q. & Sun, Y. Artificial Intelligence in Dermatology Image Analysis: Current Developments and Future Trends. en. *Journal of Clinical Medicine* **11,** 6826 (2022) (cited on page 148).

193. Liao, Q. V., Gruen, D. & Miller, S. *Questioning the AI: Informing Design Practices for Explainable AI User Experiences* in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* arXiv:2001.02478 [cs] (2020) (cited on page 208).

194. Liao, Q. V. & Varshney, K. R. *Human-Centered Explainable AI (XAI): From Algorithms to User Experiences* en. arXiv:2110.10790 [cs]. 2022. http://arxiv.org/abs/2110.10790 (2024) (cited on page 205).

195. Liker, J. K. *The Toyota Way, 14 Management Principles from the World's Greatest Manufacturer.* (McGraw-Hill, New York, 2004) (cited on pages 111, 112).

196. Lipton, Z. C., Berkowitz, J. & Elkan, C. *A Critical Review of Recurrent Neural Networks for Sequence Learning* arXiv:1506.00019 [cs]. 2015. http://arxiv.org/abs/1506.00019 (cited on page 29).

197. Liu, F., Liu, C., Zhao, L., Zhang, X., Wu, X., Xu, X., Liu, Y., Ma, C., Wei, S., He, Z., Li, J. & Yin Kwee, E. N. An Open Access Database for Evaluating the Algorithms of Electrocardiogram Rhythm and Morphology Abnormality Detection. en. *Journal of Medical Imaging and Health Informatics* **8,** 1368–1373 (2018) (cited on page 162).

198. Liu, H., Wang, Y., Fan, W., Liu, X., Li, Y., Jain, S., Liu, Y., Jain, A. & Tang, J. Trustworthy AI: A Computational Perspective. en. *ACM Transactions on Intelligent Systems and Technology* **14,** 1–59 (2023) (cited on page 38).

199. Liu, X., Wang, H., Li, Z. & Qin, L. Deep learning in ECG diagnosis: A review. en. *Knowledge-Based Systems* **227,** 107187 (2021) (cited on page 161).

200. Lomasky, R., Brodley, C. E., Aernecke, M., Walt, D. & Friedl, M. en. in *Machine Learning: ECML 2007* (eds Kok, J. N., Koronacki, J., Mantaras, R. L. D., Matwin, S., Mladenič, D. & Skowron, A.) 640–647 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007) (cited on page 31).

201. Longo, L., Brcic, M., Cabitza, F., Choi, J., Confalonieri, R., Ser, J. D., Guidotti, R., Hayashi, Y., Herrera, F., Holzinger, A., Jiang, R., Khosravi, H., Lecue, F., Malgieri, G., Páez, A., Samek, W., Schneider, J., Speith, T. & Stumpf, S. Explainable Artificial Intelligence (XAI) 2.0: A manifesto of open challenges and interdisciplinary research directions. *Information Fusion* **106,** 102301 (2024) (cited on pages 7, 233).

202. Lucieri, A., Bajwa, M. N., Alexander Braun, S., Malik, M. I., Dengel, A. & Ahmed, S. *On Interpretability of Deep Learning based Skin Lesion Classifiers using Concept Activation Vectors* in *2020 International Joint Conference on Neural Networks (IJCNN)* (Glasgow, United Kingdom, 2020), 1–10 (cited on page 186).

203. Lucieri, A., Bajwa, M. N., Dengel, A. & Ahmed, S. in *Künstliche Intelligenz im Gesundheitswesen: Entwicklungen, Beispiele und Perspektiven* (ed Pfannstiel, M. A.) 727–754 (Springer Fachmedien Wiesbaden, Wiesbaden, 2022) (cited on page 185).

204. Lundberg, S. M. & Lee, S.-I. *A Unified Approach to Interpreting Model Predictions* in (Long Beach, CA, USA, 2017) (cited on pages 44, 48, 186, 211).

205. Lwakatare, L. E., Crnkovic, I. & Bosch, J. *DevOps for AI – Challenges in Development of AI-enabled Applications* en. in *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (IEEE, Split, Croatia, 2020), 1–6. (2023) (cited on pages 82, 85).

206. Lwakatare, L. E., Crnkovic, I., Rånge, E. & Bosch, J. *From a Data Science Driven Process to a Continuous Delivery Process for Machine Learning Systems* in *Product-Focused Software Process Improvement* (eds Morisio, M., Torchiano, M. & Jedlitschka, A.) (Springer International Publishing, Cham, 2020), 185–201 (cited on pages 53, 81, 82, 87).

207. Machado, A. & Waldmann, A. *The enterprise guide to ML* 2024. https://appliedaiinitiative.notion.site/The-enterprise-guide-to-ML-a389dbdf244143f7a690b4e1980444f4 (2024) (cited on pages 4, 57).

208. MacKie MCLeod, R. *An Illustrated Guide to the Recognition of Early Malignant Melanoma* (1984) (cited on page 188).

209. Magesh, P. R., Myloth, R. D. & Tom, R. J. An Explainable Machine Learning Model for Early Detection of Parkinson's Disease using LIME on DaTSCAN Imagery. *Computers in Biology and Medicine* **126** (2020) (cited on page 187).

210. MAIF Data Scientists. *Shapash: User-friendly Explainability and Interpretability to Develop Reliable and Transparent Machine Learning Models* 2020. https://shapash.readthedocs.io/ (cited on page 210).

211. Makinen, S., Skogstrom, H., Laaksonen, E. & Mikkonen, T. *Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help?* en. in *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)* (IEEE, Madrid, Spain, 2021), 109–112. (2023) (cited on pages 5, 82).

212. Mariscal, G., Marbán, Ó. & Fernández, C. A survey of data mining and knowledge discovery process models and methodologies. en. *The Knowledge Engineering Review* **25,** 137–166. (2024) (2010) (cited on page 50).

213. Markus, A. F., Kors, J. A. & Rijnbeek, P. R. The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of Biomedical Informatics* **113** (2021) (cited on pages 205, 206).

214. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu & Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* 10.5281/zenodo.4724125. 2015. https://www.tensorflow.org/ (cited on pages 141, 145, 154, 170).

215. Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A. M. & Wagner, S. Software Engineering for AI-Based Systems: A Survey. en. *ACM Transactions on Software Engineering and Methodology* **31,** 1–59. (2023) (2022) (cited on page 5).

216. Maslej, N., Fattorini, L., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Ngo, H., Niebles, J. C., Parli, V., Shoham, Y., Wald, R., Clark, J. & Perrault, R. *The AI Index 2023 Annual Report* tech. rep. (AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, Stanford, CA, USA, 2023) (cited on page 3).

217. Mattos, D. I., Bosch, J. & Olsson, H. H. *Your System Gets Better Every Day You Use It: Towards Automated Continuous Experimentation* en. in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (IEEE, Vienna, Austria, 2017), 256–265. (2023) (cited on page 69).

218. McLennan, S., Fiske, A., Celi, L. A., Müller, R., Harder, J., Ritt, K., Haddadin, S. & Buyx, A. An embedded ethics approach for AI development. *Nature Machine Intelligence* **2,** 488–490 (2020) (cited on page 81).

219. Meske, C., Bunde, E., Schneider, J. & Gersch, M. Explainable Artificial Intelligence: Objectives, Stakeholders, and Future Research Opportunities. *Information Systems Management* **39,** 53–63 (2022) (cited on pages 206–208).

220. Meta Open Source. *React - The library for web and native user interfaces* 2013. https://react.dev/ (cited on page 143).

221. Microsoft. *What is the Team Data Science Process?* 2020. https://learn.microsoft.com/en-us/azure/architecture/data-science-process/overview (2023) (cited on pages 50, 51, 59–61, 66, 68, 102, 232).

222. Microsoft. *Model interpretability* 2024. https://learn.microsoft.com/en-us/azure/machine-learning/how-to-machine-learning-interpretability?view=azureml-api-2 (cited on page 210).

223. Microsoft Corporation. *MLOps with Azure Machine Learning - Accelerating the process of building, training, and deploying models at scale* 2021 (cited on pages 111, 112).

224. Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* **267,** 1–38 (2019) (cited on pages 7, 42, 45, 185, 203, 206).

225. MinIO Inc. *MINIO - The Object Store for AI Data Infrastructure* 2024. https://min.io/ (2024) (cited on page 166).

226. Mohseni, S., Zarei, N. & Ragan, E. D. A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems. *ACM Transactions on Interactive Intelligent Systems* **11,** 1–45 (2021) (cited on pages 205–208).

227. Molino, P. & Ré, C. *Declarative Machine Learning Systems* arXiv:2107.08148 [cs]. 2021. http://arxiv.org/abs/2107.08148 (2023) (cited on page 63).

228. Molnar, C. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable* 2nd ed. (2022) (cited on pages 43–45, 48).

229. Moskovitch, R., Nissim, N., Stopel, D., Feher, C., Englert, R. & Elovici, Y. *Improving the Detection of Unknown Computer Worms Activity Using Active Learning* in *KI 2007: Advances in Artificial Intelligence* (eds Hertzberg, J., Beetz, M. & Englert, R.) (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007), 489–493 (cited on page 32).

230. Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J. & Fernández-Leal, Á. Human-in-the-loop machine learning: a state of the art. en. *Artificial Intelligence Review* **56,** 3005–3054. (2024) (2023) (cited on page 4).

231. Mridha, K., Uddin, M. M., Shin, J., Khadka, S. & Mridha, M. F. An Interpretable Skin Cancer Classification Using Optimized Convolutional Neural Network for a Smart Healthcare System. *IEEE Access* **11,** 41003–41018 (2023) (cited on page 186).

232. Mueller, S. T., Veinott, E. S., Hoffman, R. R., Klein, G., Alam, L., Mamun, T. & Clancey, W. J. *Principles of Explanation in Human-AI Systems* en. in (2021) (cited on page 45).

233. Munappy, A. R., Mattos, D. I., Bosch, J., Olsson, H. H. & Dakkak, A. *From Ad-Hoc Data Analytics to DataOps* en. in *Proceedings of the International Conference on Software and System Processes* (ACM, Seoul Republic of Korea, 2020), 165–174 (cited on page 52).

234. Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R. & Yu, B. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences* **116,** 22071–22080 (2019) (cited on page 185).

235. Murphy, K. P. *Machine learning: a probabilistic perspective* en (MIT Press, Cambridge, MA, 2012) (cited on pages 24, 25).

236. Muslea, I., Minton, S. & Knoblock, C. A. Active Learning with Multiple Views. *Journal of Artificial Intelligence Research* **27,** 203–233 (2006) (cited on page 133).

237. Nachbar, F. & Vogt, T. The ABCD rule of dermatoscopy. en. *Journal of the American Academy of Dermatology* **30,** 551–559 (1994) (cited on pages 188, 189).

238. Nagl, S., Grünherz, V., Elia, M., Peter, T., Stieler, F., Bauer, B., Messmann, H. & Ebigbo, A. Automatische dreidimensionale Rekonstruktion des Ösophagus zur Vorhersage des Therapieerfolges bei Patienten mit Achalasie. DE. *Zeitschrift für Gastroenterologie* **61.** Publisher: Georg Thieme Verlag, KV259 (2023) (cited on page 22).

239. Nagl, S., Grünherz, V., Elia, M., Stieler, F., Peter, T., Bauer, B., Muzalyova, A., Messmann, H. & Ebigbo, A. Automatic Three-Dimensional Reconstruction of the Esophagus in Achalasia Patients undergoing POEM: A Comprehensive Assessment of Treatment Outcomes and pathophysiological Changes. EN. *Endoscopy* **56.** Publisher: Georg Thieme Verlag KG, MP085 (2024) (cited on page 22).

240. Nath, V., Yang, D., Roth, H. R. & Xu, D. en. in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022* (eds Wang, L., Dou, Q., Fletcher, P. T., Speidel, S. & Li, S.) Series Title: Lecture Notes in Computer Science, 297–308 (Springer Nature Switzerland, Cham, 2022). (2023) (cited on page 64).

241. Nauta, M., Trienes, J., Pathak, S., Nguyen, E., Peters, M., Schmitt, Y., Schlötterer, J., Van Keulen, M. & Seifert, C. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. en. *ACM Computing Surveys* **55** (2023) (cited on page 45).

242. Neves, I., Folgado, D., Santos, S., Barandas, M., Campagner, A., Ronzio, L., Cabitza, F. & Gamboa, H. Interpretable heartbeat classification using local model-agnostic explanations on ECGs. *Computers in Biology and Medicine* **133** (2021) (cited on page 211).

243. Nguyen, A., Wallace, B. & Lease, M. Combining Crowd and Expert Labels Using Decision Theoretic Active Learning. en. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* **3,** 120–129. (2023) (2015) (cited on page 81).

244. Nori, H., Jenkins, S., Koch, P. & Caruana, R. *InterpretML: A Unified Framework for Machine Learning Interpretability* arXiv:1909.09223 [cs, stat]. 2019. http://arxiv.org/abs/1909.09223 (cited on page 209).

245. Northcutt, C. G., Athalye, A. & Mueller, J. *Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks* en. in (2021) (cited on page 147).

246. O'Shea, K. & Nash, R. *An Introduction to Convolutional Neural Networks* en. arXiv:1511.08458 [cs]. 2015. http://arxiv.org/abs/1511.08458 (cited on page 30).

247. O'Leary, K. & Uchida, M. *Common Problems with Creating Machine Learning Pipelines from Existing Code* in (2020) (cited on page 139).

248. OECD. *Recommendation of the Council on Artificial Intelligence* tech. rep. ECD/LEGAL/0449 (2024) (cited on page 38).

249. Olston, C., Fiedel, N., Gorovoy, K., Harmsen, J., Lao, L., Li, F., Rajashekhar, V., Ramesh, S. & Soyke, J. *TensorFlow-Serving: Flexible, High-Performance ML Serving* en. in (Long Beach, CA, USA, 2017) (cited on pages 80, 106).

250. Pan, S. J. & Yang, Q. A Survey on Transfer Learning. en. *IEEE Transactions on Knowledge and Data Engineering* **22,** 1345–1359. (2023) (2010) (cited on page 64).

251. Passonneau, R. *Measuring Agreement on Set-Valued Items (MASI) for Semantic and Pragmatic Annotation* in *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)* (eds Calzolari, N., Choukri, K., Gangemi, A., Meagaard, B., Mariani, J., Odijk, J. & Tapias, D.) (European Language Resources Association (ELRA), Genoa, Italy, 2006) (cited on page 181).

252. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. en. *Advances in Neural Information Processing Systems* **32,** 8024–8035 (2019) (cited on pages 124, 141, 145).

253. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A. & Cournapeau, D. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011) (cited on pages 141, 145).

254. Pehamberger, H., Steiner, A. & Wolff, K. In vivo epiluminescence microscopy of pigmented skin lesions. I. Pattern analysis of pigmented skin lesions. en. *Journal of the American Academy of Dermatology* **17.** Publisher: Elsevier BV, 571–583 (1987) (cited on page 188).

255. Petsiuk, V., Das, A. & Saenko, K. *RISE: Randomized Input Sampling for Explanation of Black-box Models* in *Proceedings of the 29th British Machine Vision Conference (BMVC2018)* arXiv:1806.07421 [cs] (UK, 2018) (cited on pages 44, 46, 187).

256. Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Larochelle, H. Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). *Journal of Machine Learning Research* **22** (2019) (cited on page 6).

257. Plotly Technologies Inc. *plotly - Collaborative data science* Montréal, QC, 2015. https://plot.ly (cited on page 167).

258. Polyzotis, N., Roy, S., Whang, S. E. & Zinkevich, M. Data Lifecycle Challenges in Production Machine Learning: A Survey. en. *ACM SIGMOD Record* **47,** 17–28. (2023) (2018) (cited on pages 66, 70, 100).

259. Raghavan, H., Madani, O. & Jones, R. Active Learning with Feedback on Both Features and Instances. en. *Journal of Machine Learning Research* **7,** 1655–1686 (2006) (cited on page 31).

260. Rahman, M. S., Rivera, E., Khomh, F., Guéhéneuc, Y.-G. & Lehnert, B. *Machine Learning Software Engineering in Practice: An Industrial Case Study* en. arXiv:1906.07154 [cs]. 2019. http://arxiv.org/abs/1906.07154 (2023) (cited on page 74).

261. Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C. & Ng, A. Y. *Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks* en. arXiv:1707.01836 [cs]. 2017. http://arxiv.org/abs/1707.01836 (2024) (cited on page 161).

262. Reyna, M. A., Sadr, N., Alday, E. A. P., Gu, A., Shah, A. J., Robichaux, C., Rad, A. B., Elola, A., Seyedi, S., Ansari, S., Ghanbari, H., Li, Q., Sharma, A. & Clifford, G. D. *Will Two Do? Varying Dimensions in Electrocardiography: The PhysioNet/Computing in Cardiology Challenge 2021* en. in *2021 Computing in Cardiology (CinC)* (IEEE, Brno, Czech Republic, 2021), 1–4. (2023) (cited on pages 161, 163, 164, 182).

263. Ribeiro, M. T., Singh, S. & Guestrin, C. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier* in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* event-place: San Francisco, California, USA (Association for Computing Machinery, New York, NY, USA, 2016), 1135–1144 (cited on pages 44, 47, 48, 185, 187, 189–192, 210).

264. Rigel, D. S., Russak, J. & Friedman, R. The Evolution of Melanoma Diagnosis: 25 Years Beyond the ABCDs. *CA: A Cancer Journal for Clinicians* **60,** 301–316 (2010) (cited on page 188).

265. Riker, A. I., Zea, N. & Trinh, T. The Epidemiology, Prevention, and Detection of Melanoma. en. **10** (2010) (cited on page 148).

266. Roh, Y., Heo, G. & Whang, S. E. A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective. *IEEE Transactions on Knowledge and Data Engineering* **33,** 1328–1347 (2021) (cited on pages 59, 100).

267. Roth, G. A., Mensah, G. A., Johnson, C. O., Addolorato, G., Ammirati, E., Baddour, L. M., Barengo, N. C., Beaton, A. Z., Benjamin, E. J., Benziger, C. P., Bonny, A., Brauer, M., Brodmann, M., Cahill, T. J., Carapetis, J., Catapano, A. L., Chugh, S. S., Cooper, L. T., Coresh, J., Criqui, M., DeCleene, N., Eagle, K. A., Emmons-Bell, S., Feigin, V. L., Fernández-Solà, J., Fowkes, G., Gakidou, E., Grundy, S. M., He, F. J., Howard, G., Hu, F., Inker, L., Karthikeyan, G., Kassebaum, N., Koroshetz, W., Lavie, C., Lloyd-Jones, D., Lu, H. S., Mirijello, A., Temesgen, A. M., Mokdad, A., Moran, A. E., Muntner, P., Narula, J., Neal, B., Ntsekhe, M., Moraes de Oliveira, G., Otto, C., Owolabi, M., Pratt, M., Rajagopalan, S., Reitsma, M., Ribeiro, A. L. P., Rigotti, N., Rodgers, A., Sable, C., Shakil, S., Sliwa-Hahnle, K., Stark, B., Sundström, J., Timpel, P., Tleyjeh, I. M., Valgimigli, M., Vos, T., Whelton, P. K., Yacoub, M., Zuhlke, L., Murray, C., Fuster, V. & GBD-NHLBI-JACC Global Burden of Cardiovascular Diseases Writing Group. Global Burden of Cardiovascular Diseases and Risk Factors, 1990-2019: Update From the GBD 2019 Study. en. *Journal of the American College of Cardiology* **76.** Place: United States, 2982–3021 (2020) (cited on page 161).

268. Rouhi, R., Clausel, M., Oster, J. & Lauer, F. An Interpretable Hand-Crafted Feature-Based Model for Atrial Fibrillation Detection. *Frontiers in Physiology* **12** (2021) (cited on page 211).

269. Roy, N. & McCallum, A. *Toward Optimal Active Learning through Sampling Estimation of Error Reduction* in *Proceedings of the Eighteenth International Conference on Machine Learning* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001), 441–448 (cited on page 36).

270. Ruder, S. *An overview of gradient descent optimization algorithms* arXiv:1609.04747 [cs]. 2017 (cited on page 27).

271. Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L. & Zhong, C. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys* **16.** Publisher: The American Statistical Association, the Bernoulli Society, the Institute . . ., 1–85 (2022) (cited on page 43).

272. Ruf, P., Madan, M., Reich, C. & Ould-Abdeslam, D. Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools. en. *Applied Sciences* **11.** (2022) (2021) (cited on pages 86, 90).

273. Rule, A., Tabard, A. & Hollan, J. D. *Exploration and Explanation in Computational Notebooks* en. in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (ACM, Montreal QC Canada, 2018), 1–12 (cited on pages 89, 94).

274. Rule, A., Tabard, A. & Hollan, J. D. *Common Problems with Creating Machine Learning Pipelines from Existing Code* in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC Canada, 2020) (cited on page 5).

275. Russell, S. J., Norvig, P. & Davis, E. *Artificial intelligence: A Modern Approach* 4th ed (PEV, 2016) (cited on page 3).

276. Saeed, W. & Omlin, C. Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems* **263** (2023) (cited on page 43).

277. Sainburg, T., McInnes, L. & Gentner, T. Q. Parametric UMAP Embeddings for Representation and Semisupervised Learning. en. *Neural Computation* **33,** 2881–2907 (2021) (cited on page 173).

278. Samuel, S., Löffler, F. & König-Ries, B. *Machine Learning Pipelines: Provenance, Reproducibility and FAIR Data Principles* in *Provenance and Annotation of Data and Processes* (eds Glavic, B., Braganholo, V. & Koop, D.) (Springer International Publishing, Cham, 2021), 226–230 (cited on page 6).

279. Schallner, L., Rabold, J., Scholz, O. & Schmid, U. *Effect of Superpixel Aggregation on Explanations in LIME – A Case Study with Biological Data* in *Machine Learning and Knowledge Discovery in Databases* (eds Cellier, P. & Driessens, K.) (Springer International Publishing, Cham, 2020), 147–158 (cited on page 187).

280. Scheffer, T., Decomain, C. & Wrobel, S. *Active Hidden Markov Models for Information Extraction* in *Advances in Intelligent Data Analysis* (eds Hoffmann, F., Hand, D. J., Adams, N., Fisher, D. & Guimaraes, G.) (Springer Berlin Heidelberg, Berlin, Heidelberg, 2001), 309–318 (cited on page 36).

281. Schulz, M., Neuhaus, U., Kaufmann, J., Badura, D., Kerzel, U., Welter, F., Prothmann, M., Kühnel, S., Passlick, J., Rissler, R., Badewitz, W., Dann, D., Gröschel, A., Kloker, S., Alekozai, E. M., Felderer, M., Lanquillon, C., Brauner, D., Gölzer, P., Binder, H., Rhode, H. & Gehrke, N. *DASC-PM v1.0 - Ein Vorgehensmodell für Data-Science-Projekte* 2020. https://opendata.uni-halle.de//handle/1981185920/33065 (2024) (cited on page 51).

282. Schwaber, K. en. in *Business Object Design and Implementation* (eds Sutherland, J., Casanave, C., Miller, J., Patel, P. & Hollowell, G.) 117–134 (Springer London, London, 1997) (cited on page 49).

283. Schwaber, K. & Sutherland, J. *The Scrum Guide* (2020) (cited on page 108).

284. Schwalbe, G. & Finzel, B. A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. en. *Data Mining and Knowledge Discovery* (2023) (cited on pages 43, 209).

285. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F. & Dennison, D. *Hidden Technical Debt in Machine Learning Systems* in *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems* **2** (2015), 2503–2511 (cited on pages 4, 75–78, 86, 100, 102).

286. Sedano, T., Ralph, P. & Peraire, C. *The Product Backlog* in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (IEEE, Montreal, QC, Canada, 2019), 200–211 (cited on page 108).

287. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. & Batra, D. *Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization* in *2017 IEEE International Conference on Computer Vision (ICCV)* (IEEE, Venice, 2017), 618–626 (cited on pages 44, 46, 47, 186, 187).

288. Senapathi, M., Buchan, J. & Osman, H. *DevOps Capabilities, Practices, and Challenges: Insights from a Case Study* en. in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018* (ACM, Christchurch New Zealand, 2018), 57–67 (cited on page 52).

289. Sener, O. & Savarese, S. *ACTIVE LEARNING FOR CONVOLUTIONAL NEURAL NETWORKS: A CORE-SET APPROACH* en. in (Vancouver, Canada, 2018) (cited on pages 155, 172).

290. Serban, A., van der Blom, K., Hoos, H. & Visser, J. *Adoption and Effects of Software Engineering Best Practices in Machine Learning* in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (2020) (cited on pages 86, 99, 109, 113).

291. Settles, B. *Active Learning Literature Survey* Computer Sciences Technical Report 1648 (University of Wisconsin–Madison, 2009) (cited on pages 4, 31–36, 66).

292. Settles, B. *Closing the Loop: Fast, Interactive Semi-Supervised Annotation With Queries on Features and Instances* in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (eds Barzilay, R. & Johnson, M.) (Association for Computational Linguistics, Edinburgh, Scotland, UK., 2011), 1467–1478 (cited on page 31).

293. Seung, H. S., Opper, M. & Sompolinsky, H. *Query by Committee* in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* event-place: Pittsburgh, Pennsylvania, USA (Association for Computing Machinery, New York, NY, USA, 1992), 287–294 (cited on page 36).

294. Shannon, C. E. A Mathematical Theory of Communication. *The Bell System Technical Journal* **27,** 379–423 (1948) (cited on page 36).

295. Shapley, L. S. in *Contributions to the Theory of Games, Volume II* (eds Kuhn, H. W. & Tucker, A. W.) doi:10.1515/9781400881970-018, 307–318 (Princeton University Press, Princeton, 1953) (cited on page 48).

296. Shetty, B., Fernandes, R., Rodrigues, A. P., Chengoden, R., Bhattacharya, S. & Lakshmanna, K. Skin lesion classification of dermoscopic images using machine learning and convolutional neural network. en. *Scientific Reports* **12,** 18134 (2022) (cited on page 148).

297. Simonyan, K., Vedaldi, A. & Zisserman, A. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps* arXiv:1312.6034 [cs]. 2014. http://arxiv.org/abs/1312.6034 (cited on pages 46, 186).

298. Speith, T. *A Review of Taxonomies of Explainable Artificial Intelligence (XAI) Methods* en. in *2022 ACM Conference on Fairness, Accountability, and Transparency* (ACM, Seoul Republic of Korea, 2022), 2239–2250 (cited on pages 43, 44).

299. Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. *Striving for Simplicity: The All Convolutional Net* arXiv:1412.6806 [cs]. 2015 (cited on page 46).

300. Sridhar, V., Subramanian, S., Arteaga, D., Sundararaman, S., Roselli, D. & Talagala, N. *Model Governance: Reducing the Anarchy of Production ML* in *Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC '18)* (Boston, MA, USA, 2018), 351–357 (cited on page 107).

301. Steidl, M., Felderer, M. & Ramler, R. The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. en. *Journal of Systems and Software* **199,** 111615. (2023) (2023) (cited on pages 6, 57, 59, 66, 67, 69, 71, 72, 74, 75, 80, 86, 87, 90, 110).

302. Stieler, F. & Bauer, B. *Git Workflow for Active Learning: A Development Methodology Proposal for Data-Centric AI Projects:* in *Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering* (eds Kaindl, H., Mannion, M. & Maciaszek, L.) (SCITEPRESS - Science and Technology Publications, Prague, Czech Republic, 2023), 202–213 (cited on pages 19, 57, 85).

303. Stieler, F. & Bauer, B. in *Evaluation of Novel Approaches to Software Engineering* (eds Kaindl, H., Mannion, M. & Maciaszek, L. A.) 321–343 (Springer Nature Switzerland, Cham, 2024) (cited on pages 19, 49, 51, 57, 85).

304. Stieler, F., Elia, M., Weigell, B., Bauer, B., Kienle, P., Roth, A., Müllegger, G., Nann, M. & Dopfer, S. *LIFEDATA - A Framework for Traceable Active Learning Projects* en. in *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)* (IEEE, Hannover, Germany, 2023), 465–474. (2023) (cited on pages 20, 123, 126, 147).

305. Stieler, F., Rabe, F. & Bauer, B. *Federated medical data - how much can deep learning models benefit?* in *AMIA 2020 Virtual Clinical Informatics Converence, May 19-21* (2020) (cited on page 22).

306. Stieler, F., Rabe, F. & Bauer, B. *Towards Domain-Specific Explainable AI: Model Interpretation of a Skin Image Classifier using a Human Approach* in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (IEEE, Nashville, TN, USA, June 2021), 1802–1809. (2023) (cited on pages 18, 185).

307. Stolz, W., Hölzel, D., Riemann, A., Abmayr, W., Przetak, C., Bilek, P., Landthaler, M. & Braun-Falco, O. *Multivariate analysis of criteria given by dermatoscopy for the recognition of melanocytic lesions* in *Book of Abstracts, Fiftieth Meeting of the American Academy of Dermatology, Dallas, Tex: Dec* (1991), 7–12 (cited on pages 185, 188).

308. Storey, M.-A., Zagalsky, A., Filho, F. F., Singer, L. & German, D. M. How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. *IEEE Transactions on Software Engineering* **43,** 185–204 (2017) (cited on page 108).

309. Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S. & Mueller, K.-R. *Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology* 2021. http://arxiv.org/abs/2003.05155 (cited on pages 51, 57–61, 66, 71–74, 78, 80).

310. Sundararajan, M., Taly, A. & Yan, Q. *Axiomatic Attribution for Deep Networks* in *Proceedings of the 34th International Conference on Machine Learning* **70** (PMLR, Sydney, Australia, 2017) (cited on page 213).

311. Suresh, H., Gomez, S. R., Nam, K. K. & Satyanarayan, A. *Beyond Expertise and Roles: A Framework to Characterize the Stakeholders of Interpretable Machine Learning and their Needs* in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (ACM, Yokohama Japan, 2021) (cited on page 207).

312. Tamburri, D. A. Sustainable MLOps: Trends and Challenges. *22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC),* 17–23 (2020) (cited on pages 59, 65, 82, 85).

313. Tammaro, A. M., Matusiak, K. K., Sposito, F. A. & Casarosa, V. Data Curator's Roles and Responsibilities: An International Perspective. en. *Libri* **69,** 89–104 (2019) (cited on page 82).

314. Tang, Y.-P., Li, G.-X. & Huang, S.-J. *ALiPy: Active Learning in Python* tech. rep. arXiv: 1901.03802 (Nanjing University of Aeronautics and Astronautics, 2019). (2020) (cited on pages 124, 126).

315. Tao, C., Gao, J. & Wang, T. Testing and Quality Validation for AI Software–Perspectives, Issues, and Practices. en. *IEEE Access* **7,** 120164–120175. (2023) (2019) (cited on page 75).

316. Tharwat, A. & Schenck, W. A Survey on Active Learning: State-of-the-Art, Practical Challenges and Research Directions. en. *Mathematics* **11,** 820 (2023) (cited on pages 31–37).

317. The Kubernetes Authors. *Kubernetes Components* 2024. https://kubernetes.io/docs/concepts/overview/components/ (2024) (cited on page 166).

318. The Kubernetes Authors. *Performing a Rolling Update* 2024. https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/ (2024) (cited on page 167).

319. The Pallets Projects. *click* 2014. https://click.palletsprojects.com (cited on page 128).

320. Thiebes, S., Lins, S. & Sunyaev, A. Trustworthy artificial intelligence. en. *Electronic Markets* **31,** 447–464 (2021) (cited on page 38).

321. Thompson, C. A., Califf, M. E. & Mooney, R. J. *Active Learning for Natural Language Parsing and Information Extraction* en. in *Proceedings of the Sixteenth International Machine Learning Conference* (Bled, Slovenia, 1999), 406–414 (cited on page 32).

322. Toreini, E., Aitken, M., Coopamootoo, K., Elliott, K., Zelaya, C. G. & Van Moorsel, A. *The relationship between trust in AI and trustworthy machine learning technologies* en. in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (ACM, Barcelona Spain, 2020), 272–283 (cited on page 38).

323. Traefik Labls. *Traefik Documentation* 2024. https://doc.traefik.io/traefik/ (2024) (cited on page 166).

324. Trautwein, J. *Interpretable Machine Learning with ECG Data* Bachelorthesis (University of Augsburg, 2022) (cited on page 17).

325. Tschandl, P., Rinner, C., Apalla, Z., Argenziano, G., Codella, N., Halpern, A., Janda, M., Lallas, A., Longo, C., Malvehy, J., Paoli, J., Puig, S., Rosendahl, C., Soyer, H. P., Zalaudek, I. & Kittler, H. Human–computer collaboration for skin cancer recognition. *Nature Medicine* **26,** 1229–1234 (2020) (cited on pages 148, 194).

326. Tschandl, P., Rosendahl, C. & Kittler, H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. en. *Scientific Data* **5,** 180161 (2018) (cited on pages 148–150, 182, 193).

327. Turing, A. M. Computing Machinery and Intelligence. *Mind, New Series* **59,** 433–460 (1950) (cited on page 3).

328. Van De Leur, R. R., Bos, M. N., Taha, K., Sammani, A., Yeung, M. W., Van Duijvenboden, S., Lambiase, P. D., Hassink, R. J., Van Der Harst, P., Doevendans, P. A., Gupta, D. K. & Van Es, R. Improving explainability of deep neural network-based electrocardiogram interpretation using variational auto-encoders. en. *European Heart Journal - Digital Health* **3,** 390–404 (2022) (cited on page 215).

329. Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E. & Yu, T. scikit-image: image processing in Python. *PeerJ* **2.** Publisher: PeerJ Inc., e453 (2014) (cited on page 190).

330. Van Der Weide, T., Papadopoulos, D., Smirnov, O., Zielinski, M. & Van Kasteren, T. *Versioning for End-to-End Machine Learning Pipelines* in *Proceedings of the 1st Workshop on Data Management for End-to-End Machine Learning* (ACM, Chicago IL USA, 2017) (cited on pages 104, 106).

331. Van Noorden, R. & Perkel, J. M. AI and science: what 1,600 researchers think. en. *Nature* **621,** 672–675 (2023) (cited on page 3).

332. Vedaldi, A. & Soatto, S. *Quick Shift and Kernel Methods for Mode Seeking* in *Computer Vision – ECCV 2008* (eds Forsyth, D., Torr, P. & Zisserman, A.) (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008), 705–718 (cited on page 187).

333. Vezhnevets, A., Buhmann, J. M. & Ferrari, V. *Active learning for semantic segmentation with expected change* en. in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Providence, RI, 2012), 3162–3169 (cited on page 35).

334. Vilone, G. & Longo, L. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion* **76,** 89–106 (2021) (cited on page 43).

335. Wagner, P., Strodthoff, N., Bousseljot, R.-D., Kreiseler, D., Lunze, F. I., Samek, W. & Schaeffter, T. PTB-XL, a large publicly available electrocardiography dataset. en. *Scientific Data* **7,** 154. (2024) (2020) (cited on pages 162, 163).

336. Wang, D., Yang, Q., Abdul, A. & Lim, B. Y. *Designing Theory-Driven User-Centric Explainable AI* in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (ACM, Glasgow Scotland Uk, 2019) (cited on pages 203, 208).

337. Wang, S., Yin, Y., Wang, D., Wang, Y. & Jin, Y. Interpretability-Based Multimodal Convolutional Neural Networks for Skin Lesion Diagnosis. *IEEE Transactions on Cybernetics* **52,** 12623–12637 (2022) (cited on page 186).

338. Wang, T., Zhao, X., Lv, Q., Hu, B. & Sun, D. *Density Weighted Diversity Based Query Strategy for Active Learning* en. in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (IEEE, Dalian, China, 2021), 156–161 (cited on page 173).

339. Washizaki, H., Uchida, H., Khomh, F. & Gueheneuc, Y.-G. *Studying Software Engineering Patterns for Designing Machine Learning Systems* in *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)* (IEEE, Tokyo, Japan, 2019), 49–495 (cited on pages 104, 106).

340. Weigell, B. *MLOps in Academia: From Concept to Implementation* MA thesis (University of Augsburg, 2024) (cited on page 18).

341. Wen, D., Khan, S. M., Ji Xu, A., Ibrahim, H., Smith, L., Caballero, J., Zepeda, L., De Blas Perez, C., Denniston, A. K., Liu, X. & Matin, R. N. Characteristics of publicly available skin cancer image datasets: a systematic review. en. *The Lancet Digital Health* **4,** e64–e74 (2022) (cited on pages 148, 150).

342. Winkler, J. K., Fink, C., Toberer, F., Enk, A., Deinlein, T., Hofmann-Wellenhof, R., Thomas, L., Lallas, A., Blum, A., Stolz, W. & Haenssle, H. A. Association Between Surgical Skin Markings in Dermoscopic Images and Diagnostic Performance of a Deep Learning Convolutional Neural Network for Melanoma Recognition. *JAMA Dermatology* **155,** 1135–1141 (2019) (cited on page 186).

343. Wirth, R. & Hipp, J. CRISP-DM: Towards a Standard Process Model for Data Mining (2000) (cited on pages 50, 59, 60, 66, 73, 232).

344. Wolf, C. T. *Explainability scenarios: towards scenario-based XAI design* in *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Association for Computing Machinery, New York, NY, USA, 2019), 252–257 (cited on page 203).

345. Xiang, A. & Wang, F. *Towards Interpretable Skin Lesion Classification with Deep Learning Models* in *AMIA Annual Symposium Proceedings Archive* (2020), 1246–1255 (cited on page 187).

346. Xie, Y., Cruz, L., Heck, P. & Rellermeyer, J. S. *Systematic Mapping Study on the Machine Learning Lifecycle* en. in *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)* (IEEE, Madrid, Spain, 2021), 70–73. (2023) (cited on page 57).

347. Xu, X., Jeong, S. & Li, J. Interpretation of Electrocardiogram (ECG) Rhythm by Combined CNN and BiLSTM. en. *IEEE Access* **8,** 125380–125388 (2020) (cited on pages 170, 172, 182).

348. Xu, Z., Akella, R. & Zhang, Y. en. in *Advances in Information Retrieval* (eds Amati, G., Carpineto, C. & Romano, G.) 246–257 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007) (cited on page 36).

349. Yang, W., Le, H., Laud, T., Savarese, S. & Hoi, S. C. H. *OmniXAI: A Library for Explainable AI* arXiv:2206.01612 [cs]. 2022. http://arxiv.org/abs/2206.01612 (cited on page 210).

350. Yang, Y.-Y., Lee, S.-C., Chung, Y.-A., Wu, T.-E., Chen, S.-A. & Lin, H.-T. libact: Pool-based Active Learning in Python. arXiv: 1710.00379. (2020) (2017) (cited on page 124).

351. Yarowsky, D. *Unsupervised word sense disambiguation rivaling supervised methods* en. in *Proceedings of the 33rd annual meeting on Association for Computational Linguistics* - (Association for Computational Linguistics, Cambridge, Massachusetts, 1995), 189–196 (cited on page 33).

352. Zacharias, J., Von Zahn, M., Chen, J. & Hinz, O. Designing a feature selection method based on explainable artificial intelligence. *Electronic Markets* **32,** 2159–2184 (2022) (cited on page 218).

353. Zaharia, M. A., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F. & Zumar, C. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.* **41,** 39–45 (2018) (cited on pages 72, 79, 91, 107, 117, 144, 165).

354. Zettler, K. *Trunk-based development* 2023. https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development (2023) (cited on page 86).

355. Zha, D., Bhat, Z. P., Lai, K.-H., Yang, F., Jiang, Z., Zhong, S. & Hu, X. *Data-centric Artificial Intelligence: A Survey* en. arXiv:2303.10158 [cs]. 2023. http://arxiv.org/abs/2303.10158 (2023) (cited on pages 59, 101).

356. Zhang, D., Yang, S., Yuan, X. & Zhang, P. Interpretable deep learning for automatic diagnosis of 12-lead electrocardiogram. en. *iScience* **24,** 102373. (2022) (2021) (cited on pages 211, 213–215).

357. Zhang, J. M., Harman, M., Ma, L. & Liu, Y. *Machine Learning Testing: Survey, Landscapes and Horizons* en. arXiv:1906.10742 [cs, stat]. 2019. http://arxiv.org/abs/1906.10742 (2022) (cited on page 75).

358. Zhang, Y., Lease, M. & Wallace, B. Active Discriminative Text Representation Learning. en. *Proceedings of the AAAI Conference on Artificial Intelligence* **31** (2017) (cited on page 35).

359. Zhao, L., Sukthankar, G. & Sukthankar, R. *Incremental Relabeling for Active Learning with Noisy Crowdsourced Annotations* en. in *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing* (IEEE, Boston, MA, USA, 2011), 728–733. (2023) (cited on page 81).

360. Zhdanov, F. *Diverse mini-batch Active Learning* en. arXiv:1901.05954 [cs, stat]. 2019. http://arxiv.org/abs/1901.05954 (cited on pages 37, 173).

361. Zheng, J., Chu, H., Struppa, D., Zhang, J., Yacoub, S. M., El-Askary, H., Chang, A., Ehwerhemuepha, L., Abudayyeh, I., Barrett, A., Fu, G., Yao, H., Li, D., Guo, H. & Rakovski, C. Optimal Multi-Stage Arrhythmia Classification Approach. en. *Scientific Reports* **10,** 2898 (2020) (cited on page 162).

362. Zheng, J., Zhang, J., Danioko, S., Yao, H., Guo, H. & Rakovski, C. A 12-lead electrocardiogram database for arrhythmia research covering more than 10,000 patients. en. *Scientific Data* **7,** 48 (2020) (cited on page 162).

363. Zheng, Z. & Padmanabhan, B. *On Active Learning for Data Acquisition* in *Proceedings of the 2002 IEEE International Conference on Data Mining* (IEEE Computer Society, USA, 2002), 562 (cited on page 31).

364. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. & Torralba, A. *Learning Deep Features for Discriminative Localization* in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), 2921–2929 (cited on page 186).

365. Zhou, J., Chen, F. & Holzinger, A. in *xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers* (eds Holzinger, A., Goebel, R., Fong, R., Moon, T., Müller, K.-R. & Samek, W.) 375–386 (Springer International Publishing, Cham, 2022) (cited on page 42).

366. Zhou, Y., Yu, Y. & Ding, B. *Towards MLOps: A Case Study of ML Pipeline Platform* in *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)* (IEEE, Beijing, China, 2020), 494–500 (cited on page 53).

367. Zhu, X. Semi-Supervised Learning Literature Survey. en (2008) (cited on page 33).

368. Ziethmann, P., Stieler, F., Pfrommer, R., Schlögl-Flierl, K. & Bauer, B. *Towards a Framework for Interdisciplinary Studies in Explainable Artificial Intelligence* in *Artificial Intelligence in HCI* (eds Degen, H. & Ntoa, S.) (Springer Nature Switzerland, Cham, 2024), 316–333 (cited on pages 20, 43, 203).

369. Zinkevich, M. *Rules of Machine Learning: Best Practices for ML Engineering* 2019. `https://developers.google.com/machine-learning/guides/rules-of-ml` (2023) (cited on pages 80, 101, 102, 104, 107).

# List of Abbreviations

**A**

**Adam** Adaptive Moment Estimation. 27, 28

**AI** Artificial Intelligence. iii, iv, ix, xiv, 3–15, 17–21, 23–25, 38–43, 45, 49, 51, 53, 57, 58, 65, 67, 71, 74, 76–78, 82, 83, 85–90, 92–95, 99, 102–108, 110–116, 119, 120, 123, 145–148, 161, 185–188, 203–210, 215–220, 222–226, 229–233

**AL** Active Learning. iii, 4–11, 13, 16, 19, 20, 23, 25, 31–36, 38, 57–79, 81–83, 85, 88, 94–97, 101, 108, 115, 116, 123–128, 133, 136–140, 143–148, 150–154, 157–159, 161, 164, 167, 168, 172, 174, 177, 179, 180, 182, 204, 206–208, 222, 229–233

**API** Application Programming Interface. 117, 128, 130, 133, 137, 143–145, 180

**C**

**CAM** Class Activation Mapping. 186, 187

**CD** Continuous Delivery. 52, 53, 82, 86, 87, 91, 92, 103, 104, 108, 115, 117–120, 138, 141, 144, 151, 152, 165, 166, 209

**CDSS** Clinical Decision Support System. 161, 185, 188, 201, 210, 216, 219, 221

**CI** Continuous Integration. 52, 53, 82, 86, 87, 89, 91–93, 96, 102–105, 108, 115, 117–120, 138, 141, 144, 151, 152, 165, 166, 209

**CLI** Command Line Interface. 128–130, 137, 166

**CNN** Convolutional Neural Network. 30, 44, 148, 170, 186, 187, 210, 211

**CPU** Central Processing Unit. 151, 165

**CRISP-DM** Cross Industry Standard Process for Data Mining. 50, 51, 59, 232

**CSV** Comma-separated Values. 155, 173

**CT** Continuous Training. 53, 86, 87, 91, 92, 103, 104, 115, 117–120, 138, 141, 151, 152, 165, 166, 209

**D**

**DAG** Directed Acyclic Graph. 139, 140, 142, 144, 152, 155, 167

**DataOps** Data Operations. 51–53, 57

**DB** Database. 180

**DevOps** Development and Operations. 11, 51–53, 57, 73, 74, 82, 83

**DL** Deep Learning. 13, 28, 124, 148, 161, 186

**DNN** Deep Neural Network. 6, 161, 182, 187, 193, 201, 230, 231

**DVC** Data Version Control. 87, 90, 117, 137–139, 141, 144, 152, 159, 165, 168

**E**

**ECG** Electrocardiogram. iii, iv, xi, xiii–xv, 9, 12, 17, 117, 118, 161–165, 167–175, 179, 180, 182, 203, 210–217, 219–223, 225, 226, 230, 231

**ETL** Extract Transform Load. 65, 139, 151, 152, 164, 168

**EU** European Union. 21, 206

**F**

**FN** False Negative. 171

**FP** False Positive. 171, 188

**G**

**GB** Gigabyte. 151, 165

**GPU** Graphics Processor Unit. 68, 91, 117, 118, 138, 151, 165

**GUI** Graphical User Interface. 126, 127, 129, 143, 144, 166, 182

**H**

**HCI** Human Computer Interaction. 21, 45

**HDD** Hard Disk Drive. 151

**I**

**IBC** Imaging Biomarker Clues. 186

**IDE** Integrated Development Environments. 74

**IML** Interpretable Machine Learning. iv, 6, 7, 11, 16, 17, 23, 43–46, 203–206, 210, 212, 216, 218, 221, 226, 231, 233

**ISIC** International Skin Imaging Collaboration. 148, 150

**J**

**JSC** Jaccard Similarity Coefficient. 172

**JSON** JavaScript Object Notation. 129, 130

**JWT** JSON Web Token. 130

**K**

**KDD** Knowledge Discovery in Databases. 49, 50, 66

**KDE** Kernel Density Estimation. 198, 199

**L**

**LIME** Local Interpretable Model-Agnostic Explanation. xiv, 44, 47, 48, 185, 187, 189–192, 200, 201, 210–212, 214, 216, 218, 226, 231

**LSTM** Long Short-Term Memory. 29, 170

**M**

**MCC** Matthews Correlation Coefficient. 172

**ML** Machine Learning. iii, xiv, 3, 4, 6, 8, 9, 11–14, 16–18, 23, 24, 31, 38, 43, 51, 53, 57–64, 66–76, 78–83, 86–97, 99–107, 109–119, 123–127, 130, 136–145, 147, 148, 150–157, 161, 164–173, 175, 182, 185–188, 193, 200, 201, 204–209, 211, 212, 219, 223, 229–233

**MLOps** Machine Learning Operations. 11, 16–18, 51, 53, 57, 69, 74, 77, 82, 83, 86, 90, 110, 111, 123, 151, 164

**N**

**NAdam** Nesterov-accelerated Adaptive Moment Estimation. 28

**O**

**ORM** Object Relation Mapper. 133, 134

**Q**

**QS** Query Strategy. 9, 33–37, 62, 63, 66–68, 72, 73, 77–79, 81, 97, 124–127, 137, 141, 142, 145, 149, 153, 155, 157–160, 163, 168, 169, 172–175, 177, 179, 182, 232

**R**

**RAM** Random-Access Memory. 151, 165

**ReLU** Rectified Linear Unit. 28, 30, 170

**REST** Representational State Transfer. 130, 133, 143

**RGB** Red Green Blue. 149, 191

**RISE** Randomized Input Sampling for Explanations. 187

**RNN** Recurrent Neural Network. 29, 30, 170

**ROC** Receiver Operating Characteristic. 78, 188

**S**

**SCM** Source Control Management. 91–93, 103, 105, 113, 117, 151, 152, 164–167

**SDK** Software Development Kit. 209

**SE** Software Engineering. 5, 11, 23, 93, 99, 101, 110, 111, 113, 115, 230, 232

**SHAP** Shapley Additive Explanations. 44, 48, 186, 211–214, 216, 218, 220, 226

**SME** Small and Medium Entrepreneurs. 110

**SQL** Structured Query Language. 133

**SSD** Solid State Drive. 151, 165

**SSH** Secure Socket Shell. 117, 139

**T**

**TB** Terabyte. 151, 165

**TDSP** Team Data Science Prozess. 50, 51, 59, 61, 66, 232

**TF** TensorFlow. 126, 141, 145, 154, 170

**TN** True Negative. 171

**TP** True Positive. 171

**U**

**UI** User Interface. 123, 143, 144, 164–169, 173, 210

**UMAP** Uniform Manifold Approximation and Projection. 173

**X**

**XAI** Explainable Artificial Intelligence. iii, iv, xiv, 12, 16, 21, 23, 43, 45, 143, 185, 186, 188, 201, 203–209, 215, 217, 218, 222, 226, 231, 233