Open Source MLOps: How to Unlock the Potential of Machine Learning

Master Thesis

Malte Hedderich | 2432139
Business Informatics M.Sc.

TECHNISCHE
UNIVERSITÄT
DARMSTADT

WIRTSCHAFTS
INFORMATIK

Malte Hedderich
Matriculation Number: 2432139
Academic Program: M.Sc. Business Informatics

Master Thesis
Topic: "Open Source MLOps: How to Unlock the Potential of Machine Learning"

Submitted: 5. July 2022

Supervisor: M.Sc. Timo Sturm

Prof. Dr. Peter Buxmann
Fachgebiet Wirtschaftsinformatik | Software & Digital Business
Fachbereich Rechts- und Wirtschaftswissenschaften
Technische Universität Darmstadt
Hochschulstraße 1
64289 Darmstadt

Abstract

Machine learning continues to gain traction in the enterprise world, regardless of company size or industry. However, many companies struggle to bring machine learning models from the lab into production systems. As a result, numerous machine learning projects are either delayed or fail before reaching their final stage. While Development Operations (DevOps) practices for automating and standardizing development workflows are well established, the necessary structures for Machine Learning Operations (MLOps) are often lacking. This research begins by identifying seventeen challenges organizations typically face when adopting machine learning and fourteen best practices for overcoming them, based on a Multivocal Literature Review (MLR). In addition, sixty-three open-source tools that assist organizations in establishing MLOps capabilities are outlined. Based on this, an MLOps architecture composed entirely of open-source components is designed, validated, and implemented as a prototype for the application context of a medium-sized software company that develops AI-enabled solutions for enterprise customers. Pre-implementation validation is based on a focus group of eight experts with six distinct roles. This work thus supports organizations in building their customized MLOps solutions and uncovers prevalent practical challenges for research.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AACODS	Authority, Accuracy, Coverage, Objectivity, Date, Significance
AGPL	Affero Lesser General Public License
AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under the Curve
AWS	Amazon Web Services
CACE	Change Anything Changes Everything
CD	Continuous Delivery
CI	Continuous Integration
CFG	Confirmatory Focus Group
CLI	Command Line Interface
CRISP-DM	Cross Industry Standard Process for Data Mining
DAG	Directed Acyclic Graph
DataOps	Data Operations
DL	Deep Learning
DevOps	Development Operations
EFG	Exploratory Focus Group
ELT	Extract, Load, Transform
ETL	Extract, Transform, Load
FMDS	Foundational Methodology for Data Science
gRPC	google Remote Procedure Call
GPL	General Public License
IaaS	Infrastructure as a Service
IS	Information Systems
KDD	Knowledge Discovery in Databases
LGPL	Lesser General Public License
MLR	Multivocal Literature Review
MLaaS	Machine Learning as a Service
MLOps	Machine Learning Operations
MLSecOps	Machine Learning Security Operations
NLP	Natural Language Processing
OSS	Open-Source Software
REST	REpresentational State Transfer
ROC	Receiver Operating Characteristics

RQ	Research Question
SaaS	Software as a Service
SE	Software Engineering
SEMMA	Sample, Explore, Modify, Model, Assess
TAR	Technical Action Research
TDSP	Team Data Science Process

1 Introduction

Jeff Bezos wrote in one of his annual letters to the Amazon shareholders:

> "But much of what we do with machine learning happens beneath the surface. Machine learning drives our algorithms for demand forecasting, product search ranking, product and deals recommendations, merchandising placements, fraud detection, translations, and much more. Though less visible, much of the impact of machine learning will be of this type – quietly but meaningfully improving core operations."[1]

The quote illustrates how diverse the application areas of machine learning are and that many use cases are not directly apparent to the end-user. Additionally, the adoption of Artificial Intelligence (AI) is no longer limited to the pursuits of a few large technology companies. In a survey conducted in 2021 by McKinsey Analytics, fifty-six percent of the surveyed companies reported that they use AI in at least one business function.[2]

Despite the growing importance of AI and the increasing related budgets, the time it takes companies to bring machine learning models into production[3] systems is also increasing.[4] Furthermore, many machine learning projects do not even reach the launch stage. According to a survey conducted in 2019 with 227 respondents, seventy-eight percent of AI projects stall at some stage before deployment.[5] A different survey by Gartner (2020) found that only fifty-three percent of AI projects make it from prototypes to deployment. The significant deviation in these values indicates that it is difficult to obtain a reliable number for the market as a whole. However, it can be stated that many companies are experiencing difficulties concerning their AI projects.

Machine Learning Operations (MLOps) aims to counteract this by establishing a range of practices for building efficient and repeatable workflows.[6] At the same time, MLOps is not only about streamlining the processes for developing and deploying machine learning models but also about continuous improvement and monitoring. Thereby, MLOps assists companies in adapting faster to external changes and reducing their costs and cost variance (higher efficiency in AI).[7]

[1] Bezos (2017).

[2] Cf. Chui et al. (2021).

[3] *Production* in this context refers to a state in which the application is available to the target audience.

[4] Cf. Algorithmia (2021).

[5] Cf. Alegion (2019).

[6] Cf. Gujjar and Kumar (2022).

[7] Cf. Chui et al. (2021).

However, the substantial number of failing AI projects in conjunction with the growing time from project initiation to rollout indicates that the potential of MLOps is not yet fully exploited in practice. One apparent reason for this is the novelty of AI adoption in the wider corporate world and the associated lack of expertise within the companies. Paleyes et al. (2020) further identified a lack of deployment reports, resulting in limited knowledge transfer. Another concern for many organizations is a lack of tools for creating and managing robust machine learning pipelines.[8]

This research aims to identify the difficulties companies face when adopting machine learning models in production systems and compile best practices to overcome them. Furthermore, open-source tools to support the implementation of MLOps practices within the organization will be highlighted. Based on this, an MLOps architecture will be designed and validated in the context of a medium-sized software company[9] that develops AI-enabled solutions for enterprise customers. Finally, the improvements resulting from the validation will be incorporated and the enhanced architecture will be implemented as a prototype.

The various use cases of machine learning and the constant change in this field require continuous adaptation, which is why there can be no definitive solution suitable for every use case. Therefore, this thesis serves as an entry point for implementing customized MLOps solutions by practitioners. Furthermore, it provides the groundwork for establishing a continuous optimization process. The strict focus on open-source products was chosen to reduce dependence on individual providers and propose solutions suitable for a broad target group ranging from individual researchers to large companies. In addition, this creates the opportunity to expand or adapt individual components, as the source code is publicly available.

The overall structure of this thesis follows the design science methodology outlined by Wieringa (2014), which is based on Peffers et al. (2007) often cited design science methodology. This thesis's descriptive and design research is based on a Multivocal Literature Review (MLR), which synthesized a total of seventy-four sources. The design artifact of the MLOps architecture was validated through a focus group involving eight experts from six distinct roles and improved based on the feedback collected.

The MLR thus constitutes the engineering cycle's problem investigation and treatment design stages, as shown in Figure 1. The conducted focus group, in conjunction with the implementation of the prototype, forms the treatment validation. These three stages represent

[8] Cf. Gartner (2020).

[9] Approximately 500 employees with annual sales of around 80 million euros in 2021.

the design cycle as defined by Wieringa (2014). The implementation in the engineering cycle corresponds to the implementation in real-world operation and, therefore, cannot be realized through design science research as in this thesis. That also explains the distinction between validation and evaluation since the latter refers to implementation in the real world. Furthermore, the author used the term *treatment* instead of *solution* since the designed artifact may solve the problem only partially or not at all. However, this fine distinction is not made within this thesis since partial solutions still represent a subset of the solution space.

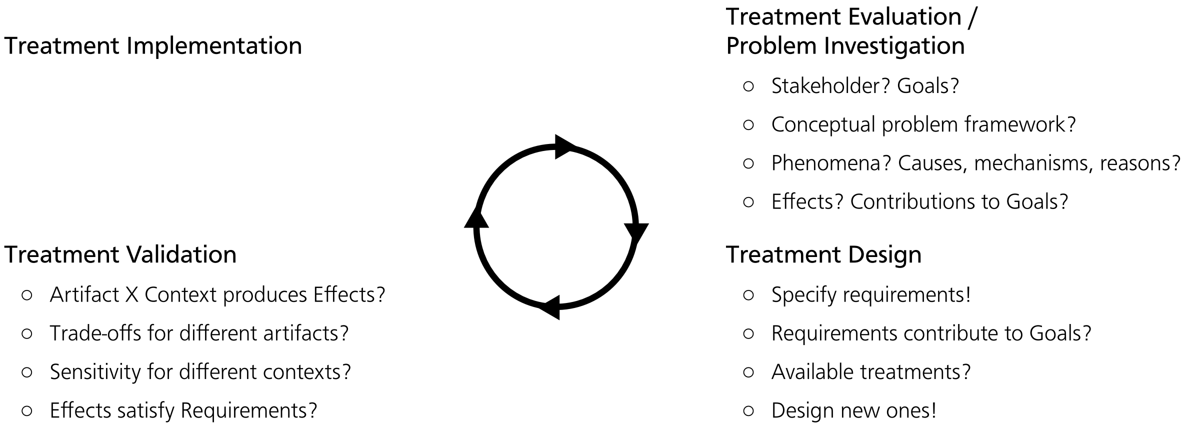

Treatment Implementation
Treatment Evaluation / Problem Investigation
○ Stakeholder? Goals?
○ Conceptual problem framework?
○ Phenomena? Causes, mechanisms, reasons?
○ Effects? Contributions to Goals?
Treatment Validation
○ Artifact X Context produces Effects?
○ Trade-offs for different artifacts?
○ Sensitivity for different contexts?
○ Effects satisfy Requirements?
Treatment Design
○ Specify requirements!
○ Requirements contribute to Goals?
○ Available treatments?
○ Design new ones!

Figure 1: The Engineering Cycle by Wieringa (2014).[10]

Any design science project begins with the problem investigation in which the problems to be addressed by the project are identified, described, explained, and evaluated.[11] Therefore, the research for this thesis begins with an in-depth assessment of the existing obstacles and challenges related to the adaption of machine learning in production systems. This investigation phase ensures that subsequent solutions do not overlook the underlying problem or the needs of stakeholders. The Research Question (RQ) chosen for this purpose is deliberately broad and aims at the entire machine learning system. Consequently, the provision of the machine learning models is usually only one part of a more extensive system. Although this inevitably implies that MLOps cannot solve all identified challenges, it is vital to examine the system holistically to avoid designing solutions that cause difficulties in other areas.

[10] The question marks indicate knowledge questions, and the exclamation marks indicate design problems.

[11] Cf. Wieringa (2014).

Accordingly, the first RQ is formulated as follows:

RQ1: What challenges typically arise with the adoption of machine learning in production systems?

The second stage is the treatment design, which aims to eliminate or mitigate the identified problems. Research evidence and industry reports must be closely examined to build on existing knowledge. This knowledge can be used to derive best practices, avoid known pitfalls, and help to develop more effective MLOps solutions. However, there is more than just existing knowledge to build on; a wide range of open-source projects that aim to facilitate the productive use of machine learning exist. These offer the option to use existing software components and enhance them according to individual demands. The second RQ of this thesis aims to bring these two kinds of resources together and thereby provide a foundation for practitioners to build their own MLOps solutions.

RQ2: Which best practices and open-source tools exist to address these challenges?

Only a limited number of the best practices and existing tools cover the entire machine learning lifecycle. Therefore, a holistic MLOps architecture is usually composed of different components. Which components suit best and how they are integrated depends on the specific use case. This is reflected in the third stage of the design cycle with the treatment validation. Validation here refers to verifying that a treatment, if implemented, would benefit the stakeholders' goals.[12] This implies that the validation always belongs to a specific problem context and cannot be generalized without further consideration.

The problem context considered in this research is that of a medium-sized software company that provides a variety of AI-enabled solutions in the form of hosted services to enterprise customers; from now on referred to as AI service provider.

In the light of the above, the final RQ of this thesis reads as follows:

RQ3: How could an effective MLOps solution be composed of the individual open-source components in the context of an AI service provider?

In summary, this thesis will provide a foundation for more targeted research by highlighting the challenges associated with the adoption of machine learning in production systems. Seventeen challenges were identified and explained in detail. In addition, the best practices will enable practitioners to more effectively design their unique MLOps solutions. Fourteen best

[12] Cf. Wieringa (2014).

practices were discussed, along with design options where possible. Furthermore, sixty-three active Open-Source Software (OSS) projects that are widely utilized to build MLOps capabilities were identified, and the most popular ones were examined in more detail. From these, an MLOps architecture based entirely on open-source components was designed and validated in the context of an AI service provider. Finally, it was implemented as a prototype after enhancing the architecture based on expert feedback. The prototypical implementation represents an additional validation step, particularly regarding the components' interaction.

This thesis is divided into the following chapters with their respective contributions:

Chapter 2: Provides the necessary fundamentals. This involves explaining the most important technical terms and process models from data science and MLOps.

Chapter 3: Serves to illustrate the methodology of this research and underpins the quality of the results. Accordingly, MLRs and focus groups are presented as well as the rationale for choosing these methods.

Chapter 4: Gives an overview of the challenges related to the adoption of machine learning within production systems. The specification of the issues supports the target-oriented search for solutions.

Chapter 5: Provides an overview of established best practices and open-source tools in the MLOps field, thus enabling practitioners to build on established approaches. This chapter also discusses each best practice in relation to the identified challenges.

Chapter 6: Demonstrates an MLOps architecture that has been validated in a practical use case and thereby illustrates how the individual components can be integrated. This section also explains the reasons for choosing the individual open-source components and the modifications resulting from the practical case validation.

Chapter 7: Provides insights into the implemented architecture and its derived limitations and thus assists in understanding the interaction of the components.

Chapter 8: Discusses the underlying meaning of the results in the context of the current research and the validity of the results systematically. This allows the contextualization of this research and supports its credibility.

Chapter 9: Summarizes the main findings along with their implications, limitations, and potential for future research. This includes a discussion of this work's practical and theoretical contributions.

2 Theoretical Background

The first section provides a general understanding of the basic terminology before examining the open-source topic in more detail. Subsequently, it is necessary to understand the basic process of machine learning projects to understand this work and the topic of MLOps. For this purpose, the most common data science process models are described. In particular, the Cross Industry Standard Process for Data Mining (CRISP-DM) is explained in more detail, as it is still widely used in the industry.[13] The conceptual distinction between data mining and data science is also discussed, and essential changes in more recent process models are highlighted. Even though these ideal-typical processes often require adaptions in practice due to the diverse use cases of machine learning, they provide a basic understanding necessary for the following MLOps-specific explanations. This will begin with a general definition of MLOps before discussing the individual process stages in the MLOps lifecycle and the MLOps core capabilities. It also addresses the relation between Development Operations (DevOps) and MLOps.

2.1 Terminology

The following terminology is used throughout this thesis or contributes to the general understanding of the subsequent elaborations.

Artificial Intelligence (AI): In computer science, AI describes the concept of intelligent agents, i.e., anything that can perceive their environment and direct their actions to maximize the probability of success for a given goal.[14]

Machine Learning: A subfield of AI in which computers learn based on experience rather than following explicit instructions.[15] Deep Learning (DL) is a branch of machine learning where depth refers to the number of composition levels of non-linear operations in the learned function.[16] However, many current machine learning systems still utilize shallow architectures with one, two, or three levels, as Bengio (2009) stated.

[13] See Martinez-Plumed et al. (2021).

[14] Cf. Russell and Norvig (2016).

[15] Cf. Samuel (1959).

[16] Cf. Bengio (2009).

Supervised Learning: A subfield of machine learning in which input-output pairs are used to learn the function that maps input to output.[17]

Unsupervised Learning: A subfield of machine learning in which patterns are learned based on inputs without explicit outputs.[18] The most common approach is clustering, where the input data is assigned to potentially useful clusters of similar data.

Semi-Supervised Learning: A subfield of machine learning in which the outputs are only available for a subset of the dataset.[19]

Reinforcement Learning: A subfield of machine learning in which learning is accomplished through various trial runs with rewards and punishments.[20]

Label: The output associated with a given input; determined either by the prediction of a machine learning model or given as the correct answer in the training data.[21]

Model: Within the context of machine learning, the term model refers to the model artifact created by the training process.[22] The model can make predictions about input data, for which it returns a label as output.

Instance: A single unit of input data.[23] In the case of tabular data, one instance represents one row. Synonymously used terms are record, example, observation, and row.

Feature: The available raw data is often not well suited for model training and must first be transformed appropriately.[24] The resulting representation is referred to as a feature. For instance, a date field could be transformed into the features such as day, month, year, calendar week, and weekday.

Pipeline: In the machine learning context, pipelines or machine learning pipelines refer to connected sequences of operations such as preprocessing, feature transformations, and model training.[25] Pipelines are thereby not only used for model training but also for data orchestration or in the form of prediction pipelines that link the feature transformation steps with the model

[17] Cf. Russell and Norvig (2016).

[18] Cf. Russell and Norvig (2016).

[19] Cf. Russell and Norvig (2016).

[20] Cf. Russell and Norvig (2016).

[21] See Russell and Norvig (2016).

[22] Cf. Amazon Web Services (2016).

[23] Cf. Amazon Web Services (2016).

[24] Cf. Amazon Web Services (2016).

[25] Cf. Schelter et al. (2017).

prediction. In addition, more complex structures in the form of Directed Acyclic Graphs (DAGs) are also subsumed under the term pipeline.

2.2 Open-Source Software

OSS is a generic term for software whose license model meets the criteria of the open-source definition.[26] In addition to open disclosure of the source code, the ten criteria include free redistribution, permission for modifications and derivations, and free use in all fields of endeavor. However, the established OSS licenses differ in how strictly they interpret the fundamental principles defined in the criteria. In addition to information obligations regarding licenses, copyrights, and modifications to the software, they differ in particular in terms of the copyleft effect.[27] The copyleft effect refers to the requirement to license software derivatives under the same license. It is included in licenses such as the General Public License (GPL), the Lesser GPL (LGPL), and the Affero GPL (AGPL). However, more than half of the OSS published on GitHub are licensed with licenses that do not have a copyleft clause, such as MIT, Apache 2.0, and the BSD 3-Clause license.[28]

Besides many individuals, companies often participate in the ongoing development of OSS projects. The motivation for the contributions of companies comes from selling complementary services, building more excellent innovation capability, and cost reduction by leveraging open-source communities.[29]

In addition to contributing to the further development of OSS, many companies use OSS. For instance, a recent study by Gentemann et al. (2021) indicates that seven out of ten German companies use OSS. The primary motivations for using OSS are cost savings, access to the source code, the prevention of vendor lock-ins, and the option of customization.[30]

2.3 The Data Science Process

MLOps is not an end in itself but aims to streamline the development and operation of machine learning models in production. An important application area for machine learning is data

[26] Cf. Open Source Initiative (2007).

[27] Cf. Schoettle (2019).

[28] See Balter (2015).

[29] Cf. Andersen-Gott et al. (2012).

[30] Cf. Balter (2015).

mining, in which algorithms find patterns in large amounts of data.[31] The patterns obtained through data mining can then be interpreted to generate knowledge from the data, as shown in the process of Knowledge Discovery in Databases (KDD) illustrated in Figure 2. It assists in understanding the data processing procedure, which is often applied similarly in modern machine learning projects.

First, the relevant subset is extracted from the data pool, outliers and noise are removed in the data pre-processing, and dimension reductions are applied in the transformation phase.[32] Which measures are suitable in the respective stages depends on the specific use case.

The KDD model is the foundation for various other data mining process models such as Sample, Explore, Modify, Model, Assess (SEMMA), and the CRISP-DM.

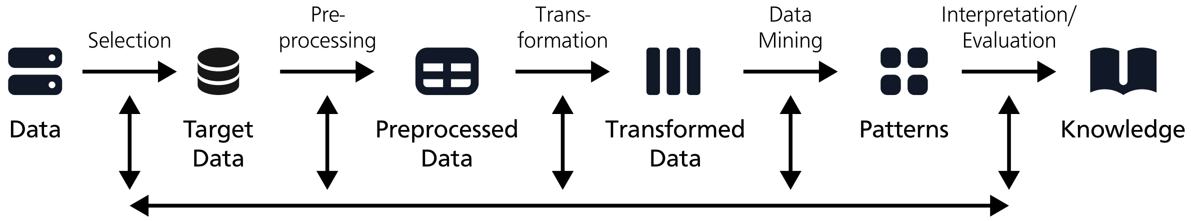


Figure 2: KDD Process by Fayyad et al. (1996b)

Since the introduction of these models more than two decades ago, data mining has evolved to include exploratory elements, which is why it is now more commonly referred to as data science.[33] Exploratory elements denote the option of extracting the value-adding components from the available data instead of defining them in advance. As a result, various new process models were developed to reflect these changes, often based on the CRISP-DM components or extending them.[34] In the following, the CRISP-DM model shown in Figure 3 is examined in more detail, as it remains the de facto standard for data science projects.[35]

The CRISP-DM model is more comprehensive than the KDD model because business understanding precedes the actual data processing, and deployment is incorporated after the evaluation. It thus considers data mining as an integrated process, while it is only one step in the KDD model.

[31] Cf. Fayyad et al. (1996a).

[32] Cf. Fayyad et al. (1996b).

[33] Cf. Martinez-Plumed et al. (2021).

[34] E.g., Ahmed et al. (2018); Cios and Kurgan (2004); Jorge et al. (2002).

[35] Cf. Martinez-Plumed et al. (2021).

The following paragraphs in which the distinct stages of the CRISP-DM model are described are all based on the initial publication by Shearer (2000) unless another source is referenced.

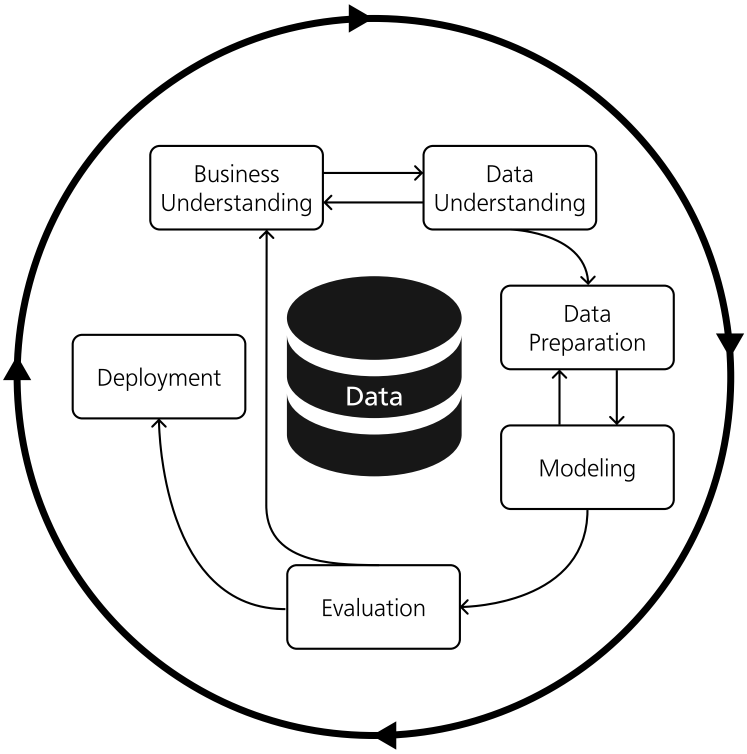


Figure 3: CRISP-DM by Shearer (2000)

The *Business Understanding* stage is about identifying the explicit goal from the business perspective to define data mining goals based on that, which can be achieved with the available data. It also includes creating a preliminary project plan for the entire data mining project.

The *Data Understanding* phase begins with the initial collection, which might involve multiple data sources. The next step is a cursory analysis of the collected data, where the format, number of records, column labels, and other surface features are reviewed to ensure that the collected data meets the relevant requirements to achieve the predefined objectives. In the following, aggregation and visualization steps are conducted, which give a first insight into the data and enable the formulation of initial hypotheses. To conclude this stage, the data quality is verified by identifying missing values, typing errors, and outliers.

The *Data Preparation* phase aims to transform the data into a suitable structure for processing by the modeling techniques. For this purpose, the first step is to analyze which attributes are irrelevant to the data mining objectives or should be excluded for other reasons such as poor data quality or technical limitations. It is crucial to justify why individual attributes are included or excluded, as this significantly influences the modeling results. The quality deficiencies identified in the previous phase must also be addressed to avoid negative influences on the modeling. This may require an exclusion or more ambitious procedures such as estimating missing values through appropriate analytical techniques. Subsequently, the clean data is used to determine the necessary or valuable transformations for the modeling. For example, it may be essential to convert categorical data into numerical values or perform mathematical operations such as determining areas from length and width or normalizing attributes. The numerical representation of an aspect of the raw data is called a feature in the context of machine learning.[36] Data integration can also be performed here, combining data from various sources into one dataset. The last step is to convert the dataset into a suitable format for the modeling technique.

The *Modeling* stage begins with selecting modeling techniques, as several would usually be applicable for the given objective. Appropriate machine learning techniques such as decision trees or convolutional neural networks might be chosen for modeling. Often several techniques are evaluated to select the best technique based on a preliminary evaluation. If the current data preparation does not fulfill the requirements of the chosen modeling techniques, a further data preparation step can take place. In the case of supervised learning techniques, where the target variable is known, the dataset is often split into a training and a test split. After a suitable test design has been defined, the modeling of one or more models begins. This is where the chosen modeling techniques are applied, and hyperparameters are optimized. Hyperparameters are parameters in learning systems that are not changed by model training but must be determined beforehand.[37] The models are then assessed by the data mining analysts based on their domain knowledge, the data mining objectives, and the defined test cases. Close cooperation with business analysts and domain experts is necessary to interpret the results in the business context. Domain expert refers to someone who is genuinely knowledgeable about the domain in which machine learning is applied but has little knowledge of machine learning.[38] The assessment should result in a ranking of modeling techniques with specified hyperparameters for this particular use case.

A more detailed evaluation of the model favored by the ranking follows in the E*valuation* stage. While the previous evaluations were based on factors such as the accuracy or generalization of the model, this evaluation focuses on the fulfillment of the business goals. The aim is to ensure

[36] Cf. Zheng and Casari (2018).

[37] Cf. Andonie (2019).

[38] Cf. Karmaker et al. (2022).

that the model sufficiently considers all essential business objectives. This involves checking whether the model meets the criteria of the business objective, but initial tests can also be conducted in real applications. In addition, it is carefully reviewed whether the model was created correctly, and all data used for this purpose are also available after the upcoming deployment. At the end of the evaluation, the project leader must decide whether the model should be deployed as it is or whether more iterations of the previous stages are necessary.

The complexity of the *Deployment* stage depends on the defined requirements. It can be limited to the generation of a report but may also be the implementation of a repeatable data mining process within a corporation. Even if the data analyst does often not conduct the actual deployment himself, a strategy for the deployment must be developed. It is also essential to monitor the model in productive operation and maintain it if necessary. In the last step, a project report describing all the deliverables and summarizing the results is produced. Furthermore, it is crucial to record the project's failures and successes to learn from them for future projects.

While the stages defined by the CRISP-DM model have remained largely unchanged in corporate practice, one crucial modification in more recent process models is important to highlight. As such, both the Foundational Methodology for Data Science (FMDS), published by IBM (2015), and the Team Data Science Process Model (TDSP) by Microsoft (2017) involve a reverse link from deployment to modeling. This refers to the generation of new training data through end-user feedback, which is a key element of the continuous training process described later in this thesis. The models can be found in the appendix in Figure 20: TDSP by Microsoft (2017) and Figure 21: FMDS by IBM (2015).

2.4 Machine Learning Operations (MLOps)

The previous subchapter illustrates that modeling in machine learning systems only occupies a small part in developing machine learning systems. The time and effort required for data collection, data cleaning, and extracting features relevant for modeling are usually more extensive.[39] Furthermore, the productive use of machine learning models requires solutions for the serving infrastructure and continuous monitoring with the option for model adjustments. Figure 4 demonstrates the multidimensionality of machine learning in production systems.

This is further complicated by the fact that machine learning systems are intertwined systems in which changes in individual areas may significantly impact the entire system. Imagine a

[39] Cf. Sculley et al. (2015).

system that uses features $x_1 \dots x_n$ as inputs for a model; if the distribution of the input value x_1 changes, the importance and weights of the other $n - 1$ features may also change.[40] Adding feature x_{n+1} or removing any feature x_j may have similar effects. The input data is never fully independent. This principle is called Change Anything Changes Everything (CACE). It applies not only to the input values but also to hyperparameters, learning settings, sampling methods, convergence thresholds, data selection, and any other control.

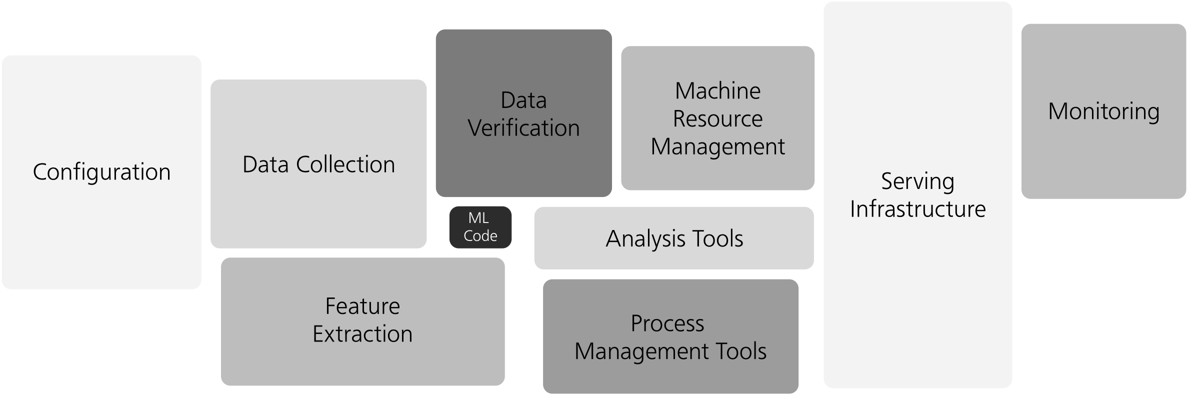


Figure 4: Elements of Machine Learning Systems by Sculley et al. (2015)

MLOps aims to address this complexity by formalizing and, where beneficial, automating the critical process stages associated with machine learning systems. It defines a set of standardized procedures and technologies to make machine learning models' development, deployment, and operation faster and more reliable. MLOps practices can lead to shorter development cycles and thus shorter time to market, better collaboration between teams, increased stability, performance, scalability, and security, and simplified operationalization and governance processes.[41] This often results in an increased return on investment from machine learning projects.

[40] Cf. Sculley et al. (2015).

[41] Cf. Salama et al. (2021).

2.4.1 MLOps Relation to DevOps

DevOps is defined by authors Jabbari et al. (2016) in a systematic mapping study of forty-nine primary sources as follows:

> "DevOps is a development methodology aimed at bridging the gap between Development (Dev) and Operations (Ops), emphasizing communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilizing a set of development practices."

MLOps pursues a similar goal by connecting machine learning development with operations, as the wording suggests. However, machine learning models are often utilized in existing or newly developed systems, demanding close collaboration and communication between the respective development and operations teams.

Continuous Integration (CI) refers to a typically automatically triggered process of interrelated steps such as compiling code, executing unit and acceptance tests, validating test coverage, verifying that coding quality standards are met, and building deployment packages.[42] Machine learning systems benefit from similar procedures, with some crucial additions since the data must be considered, and the models must be versioned in addition to the model code. This requires additional steps for data cleaning, versioning, and the orchestration of data pipelines.[43] Therefore, quality assurance must be extended to ensure compliance regarding the data used.

Continuous Delivery (CD) describes a practice of enabling the ability to deploy automatically at any point in time in a specific environment, which may not necessarily be that of the end-user.[44] This includes not only the actual software and its direct dependencies but also database upgrades or downgrades, as well as network and firewall configurations.[45] MLOps adds new requirements to established DevOps practices, as it must consider changes in the data and the model in addition to the former requirements such as resilience, queries per second, and load balancing.[46]

DevOps can thus be seen as an essential foundation for building MLOps, as many proven practices also support the MLOps process. This is illustrated by Gift and Deza (2021) in their Machine Learning Engineering Hierarchy of Needs, which is shown in Figure 5.

[42] Cf. Fitzgerald and Stol (2017).

[43] Cf. Lakshmanan et al. (2020).

[44] Cf. Fitzgerald and Stol (2017).

[45] Cf. Humble and Farley (2011).

[46] Cf. Salama et al. (2021).

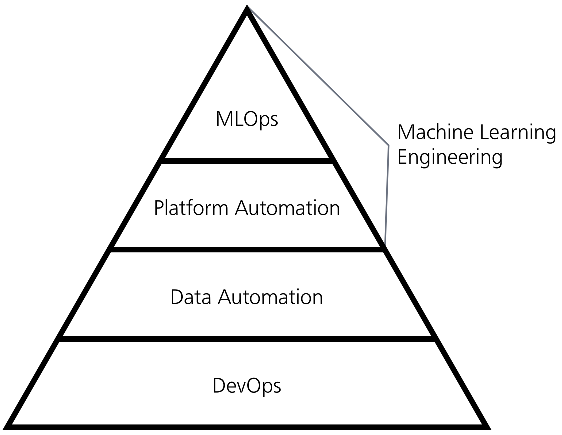


Figure 5: Machine Learning Engineering Hierarchy of Needs by Gift and Deza (2021)

The term Data Operations (DataOps) is often used in conjunction with the component *Data Automation*. DataOps describes the fundamental approach of introducing automation into data collection, validation, and verification processes.[47] Other definitions take a broader view to include transformation steps through Extract, Transform, Load (ETL) / Extract, Load, Transform (ELT) pipelines.[48]

Platform Automation aims to build structures for repeatability, scalability, and operationalization of machine learning models.[49] The authors thus define platform automation as the underlying infrastructure for MLOps. However, this fine distinction is not widespread in the literature, which is why this thesis takes a more holistic view of MLOps by considering platform automation as an integral part of it.[50]

In summary, MLOps cannot be pursued in isolation from DevOps and DataOps; it builds on them and aims to leverage established practices for the machine learning' specific characteristics.

[47] Cf. Munappy et al. (2020).

[48] See Mainali et al. (2021).

[49] Cf. Gift and Deza (2021).

[50] See, e.g., Granlund et al. (2021); Ruf et al. (2021); Salama et al. (2021).

2.4.2 MLOps Lifecycle

The MLOps Lifecycle was developed by a team from Google Cloud and represents the processes that are enabled or facilitated by MLOps. The model is illustrated in Figure 6 and comprises six complementary processes surrounding data and model management as fundamental components.

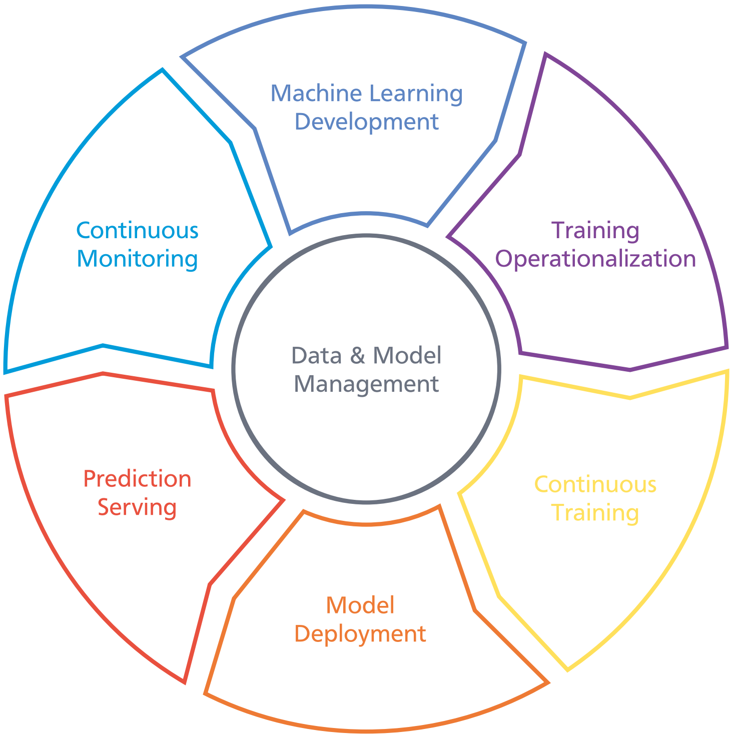


Figure 6: The MLOps Lifecycle by Salama et al. (2021)

The following explanations of the model are all based on the initial publication of Salama et al. (2021) unless another source is cited.

The process begins with *Machine Learning Development*, which focuses on building robust and repeatable model training pipelines. The pipelines consist of the stages described in subchapter 2.3: data preparation, data transformation, model training (modeling), and evaluation. The specific definitions of these stages are derived from experiments.

Training Operationalization addresses the automated and reliable packaging, testing, and deployment of the developed training pipelines.

Continuous Training refers to the repeated execution of these training pipelines with new data or changes in the model definition described by the code. Apart from manual execution, it can also be triggered by certain events, e.g., the acquisition of a specified amount of data or by a schedule. If the execution of the pipeline produces a valid model candidate, this candidate is registered in the model management.

Model Deployment covers the process of packaging, testing, and deploying models in environments for online experimentation or production serving. Not all model candidates get deployed, a selection is made from the available candidates, and the selected candidate is reviewed and approved for deployment.

Prediction Serving is concerned with serving the models used in production for inference. This means establishing Application Programming Interfaces (APIs) for secure and optimized inferences for the business's specific cloud or on-premises infrastructure.[51] According to the use case, the serving can be performed as online, batch, or streaming prediction. Online predictions refer to single requests immediately served with low latency, whereas batch prediction describes the asynchronous processing of several requests.[52] Streaming is especially prevalent in big data processing, where volume and velocity make online and offline serving unfeasible.[53]

Continuous Monitoring refers to the ongoing monitoring of the models' effectiveness and efficiency. Effectiveness in this context means the quality of model predictions, while efficiency describes hardware utilization, latency, throughput, and error rates.

Data and Model Management is the central cross-cutting function for machine learning models, providing auditability, traceability, and compliance. It also supports the shareability, reusability, and discoverability of machine-learning content.

2.4.3 Core MLOps Technical Capabilities

The comprehensive implementation of the MLOps Lifecycle poses various technical demands on enterprises. Figure 7 illustrates the core capabilities required for end-to-end MLOps implementations. The capabilities are abstracted as functional components and can be accomplished differently.[54] There are integrated machine learning platforms that attempt to cover the entire lifecycle. Nevertheless, combining products from different vendors is common to pick the most suitable product for each task. Some companies even develop their own

[51] Cf. Algorithmia (2020).

[52] Cf. Gujarati et al. (2017).

[53] Cf. Fan and Bifet (2013).

[54] Cf. Salama et al. (2021).

solutions tailored to their specific requirements. Building these capabilities is typically stepwise and should align with the organization's priorities and existing technical competencies.[55]

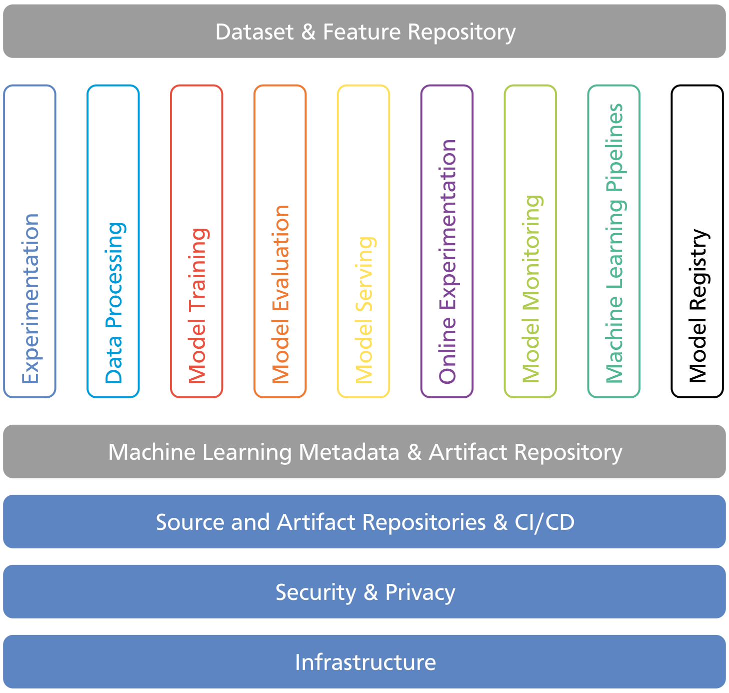


Figure 7: Core MLOps Technical Capabilities by Salama et al. (2021)

The following explanations of the core capabilities are all based on the initial publication of Salama et al. (2021) unless another source is cited.

The lower levels form the foundation for the MLOps core capabilities and are not specific to machine learning systems. Thus, many companies already have an existing cloud or on-premise *Infrastructure*, which ensures *Security & Privacy* and can be leveraged for MLOps. Furthermore, *Source and Artifact Repositories & CI/CD* solutions are often already in place due to prior investments in DevOps practices.

Building on these, the core capabilities are illustrated, with *Machine Learning Metadata & Artifact Repository* and *Dataset & Feature Repository* representing cross-cutting capabilities to enable integration and interaction across the remaining capabilities.

The *Machine Learning Metadata & Artifact Repository* is fundamental for integrating all MLOps capabilities as they consume and produce machine-learning metadata and artifacts. Examples of artifacts that arise in the machine learning context are descriptive statistics, data schemas,

[55] Cf. Salama et al. (2021).

trained models, and evaluation results. Metadata is information about these artifacts, such as their storage location, types, properties, and linkages to experiments and pipeline executions.

The *Dataset & Feature Repository* capability standardizes how data used for machine learning is defined and stored. This aims at reducing the time and effort for data preparation and feature engineering by providing reusable and shareable datasets and features. Feature engineering is the process of constructing new features from existing features to improve the predictions of the models.[56] The methods used for this depend on the data type, for example, a simple breakdown of date fields into the component's day, month, and year, but also more complex approaches like the application of principal component analysis.[57]

The *Experimentation* capability enables all machine learning developers to perform exploratory data analysis, prototype model architectures, and implement training routines. This includes the capability to track the experiments by storing information about the data used, hyperparameters, and evaluation metrics for comparison and reproducibility. The environment for this should also allow them to write modular, reusable, and testable source code that is version controlled.

The *Data Process*ing capability provides structures for preparing and transforming large datasets for scaling in machine learning development, continuous training, and prediction serving. This includes providing data connectors to numerous data sources as well as encoders and decoders for various data structures and formats.

The *Model Training* capability allows time and cost-effective training of machine learning models and the ability to scale with model and dataset size. Thereby, standard machine learning frameworks and customized runtimes should be supported, and efficient hyperparameter optimization options should be available.

The *Model Evaluation* capability enables the evaluation of model effectiveness interactively during experimentation and automatically in production. This includes calculating custom evaluation metrics on different data slices and visualizing performance differences between different models.

The *Model Serving* capability facilitates the deployment and serving of models in production environments. This incorporates options for low-latency online prediction and high-throughput offline predictions. Also, composite predictions where several models are executed

[56] Cf. Khurana et al. (2016).

[57] Cf. Zheng and Casari (2018).

hierarchically or simultaneously to aggregate the results with arbitrary preprocessing and postprocessing steps are covered here.

The *Online Experimentation* capability allows comparing the effectiveness of newly trained models in the production environment with the existing ones before they are released to production. To facilitate comparison between different models, the results of the online experiments should be integrated into the Model Registry.

The *Model Monitoring* capability is utilized to track the efficiency and effectiveness of models in production as defined in the continuous monitoring paragraph of the previous section. This allows the detection of changes in the production system, such as concept drifts. Concept drift refers to supervised learning environments where the relationship between the input data and the target variable shifts over time.[58]

The *Machine Learning Pipeline* capability enables the orchestration and execution of complex machine learning training and prediction pipelines in production. The flow of these pipelines should be automated and allow for triggering on demand, on schedule, or based on specific events.

The *Model Registry* capability provides a central location for storing machine learning models and their metadata. The data stored with the models can be technical data such as runtime dependencies, but also documentation on how the models can be used and evaluation results.

As mentioned at the beginning, these capabilities are rarely built up simultaneously. Instead, it is a process in which priorities must be set according to the company's individual business goals and resources. It is important to emphasize that almost all these capabilities are interwoven so that the implementation of one capability facilitates the implementation of the others.

2.4.4 MLOps Relation to AutoML

Automated Machine Learning (AutoML) aims to make machine learning accessible to non-machine learning experts by automating steps in the data science process.[59] However, existing tools are limited to data preparation, modeling, and evaluation tasks, as Karmaker et al. (2022) stated. For this purpose, many models are trained and the most promising one is selected, making AutoML methods much more computationally intensive than classical machine learning

[58] Cf. Gama et al. (2014).

[59] Cf. Karmaker et al. (2022).

techniques.[60] Symeonidis et al. (2022) further indicate that AutoML tools offer significantly less flexibility due to their abstractions and are thus unsuitable for many specialized tasks. However, for those cases where AutoML is suitable, it accelerates and simplifies the model creation process. Accordingly, AutoML tools can be a valuable complement to MLOps, but cannot replace manually developed machine learning pipelines for all use cases. Furthermore, other MLOps capabilities such as model monitoring, model serving, and online experimentation are not addressed by existing AutoML tools.

[60] Cf. Symeonidis et al. (2022).

3 Research Methodology

An MLR was conducted to determine what challenges typically arise when adopting machine learning in production systems (RQ 1) and how they can be overcome in practice (RQ 2). The procedure of the MLR followed the guidelines by Garousi et al. (2019), which can be found in Table 12 in the appendix. The option of addressing these issues through empirical research was discarded to include a broader range of perspectives in the time available. Due to the cross-industry relevance of MLOps in combination with the media interest in machine learning, it can be assumed that an extensive knowledge base already exists.

In contrast, the third RQ concerns a design developed in this thesis, which is why an empirical investigation is necessary for a comprehensive validation. The validation of the design is an essential step in every design science project and significantly contributes to the improvement of the design.[61] In this thesis, a focus group session was conducted for validation since the experts cannot interact with each other in individual interviews, and such interaction may yield valuable insights. The experts' interaction allows departments' conflicting interests to be identified and treated systematically. Other more time-intensive options, such as Technical Action Research (TAR), were discarded in favor of a broader perspective on the other parts of this thesis. The following explanations of the conducted focus group are more extensive than those of the MLR. This is because the procedure for MLRs follows a scientifically established process that is not very controversial. Concerning the design of focus groups, by contrast, there are significant discrepancies in the scientific literature that require consideration.

3.1 Multivocal Literature Review

While only peer-reviewed sources are usually used within systematic literature reviews, MLRs complement the white sources with grey literature. Non-multivocal literature reviews thus have the advantage of ensuring a remarkably high source quality. However, in cases where developments do not originate from academic research but the industry, important insights may be overlooked.

The importance of including scientifically unpublished industry sources in this research is already evident from the formulation of the RQs, as these address challenges from the industry. Therefore, the guidance for deciding whether to include gray literature will not be discussed

[61] Cf. Hevner et al. (2004).

here. However, the decision framework of the authors of the applied MLR guidelines can be found in Table 13 in the appendix for a comprehensive picture.

Nevertheless, including gray literature requires a more rigorous process to meet scientific standards. The following subsections explain the relevant procedures applied in this research in detail.

3.1.1 Search Process

The search process between the formal sources and the gray literature differs mainly in the search engines, as the syntactic requirements are similar. For the scientific literature, Google Scholar[62] was primarily used since its coverage already includes a wide range of relevant databases. The queries were additionally echoed with the search engines of ACM digital library[63], IEEE Xplore[64] , and Web of Science[65] to ensure no relevant results were overlooked. This led to additional results in some cases, especially for recent publications that were not yet indexed in Google Scholar. For the grey literature, Google Search[66] was used and supplemented with the search engine of arXiv[67]. For each query, only the first one hundred search results were considered initially and then expanded if the last results still yielded relevant new insights. This procedure is known as theoretical saturation.[68]

The search was initiated with the search strings *“Machine Learning” AND “Challenges” AND “Production”* and *“MLOps” AND (“Components” OR “Patterns” OR “Best Practices”)* and iteratively expanded in the search process. The decision to write out Machine Learning was made to avoid erroneous results due to confusion of the abbreviation ML with the volume unit milliliter or the programming language Meta Language.

Furthermore, forward and backward snowballing was applied to both formal and grey literature to ensure that all relevant publications were included. In the context of formal literature, this was accomplished by considering the reference lists of the literature already included (backward snowballing) and the papers referencing the included literature (forward

[62] https://scholar.google.de

[63] https://dl.acm.org

[64] https://ieeexplore.ieee.org

[65] https://www.webofscience.com

[66] https://www.google.com

[67] https://arxiv.org

[68] Cf. Garousi et al. (2019).

snowballing).[69] The concept was adapted for grey literature without references by considering HTML links from the documents and links leading to the document identified by the backlink checker ahrefs[70]. A breakdown of the source counts after each filtering step can be found in Table 16 within the search protocol in the appendix.

3.1.2 Source Selection and Quality Assessment

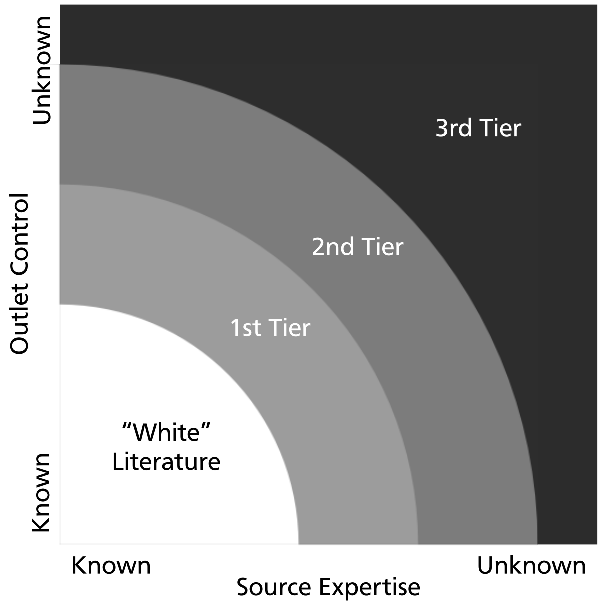
Only sources published in English or German in 2017 or later were considered within this research. The restriction to the last five years aims to ensure that only ongoing challenges and solutions are considered. The further selection process based on title, abstract, and content excluded all publications that referred to other semantic meanings of the search terms, hardware challenges, or findings that only apply to closed-source solutions.

The exclusion of qualitative inferior grey literature was based on the quality assessment checklist of grey literature for Software Engineering (SE) by Garousi et al. (2019), which can be found in Table 14 in the appendix. The checklist consists of nineteen questions that provide insights on authority, methodology, objectivity, recency, traceability, novelty, and impact, as well as a classification of the outlet type. It is based on the widely used *Authority, Accuracy, Coverage, Objectivity, Date, Significance (AACODS)* checklist but has been adapted to the SE domain.[71] The outlet types were classified based on the Shades of Grey Literature by Adams et al. (2017), as shown in Figure 8, which classifies the quality based on the expected expertise and outlet control. The model was supplemented with media types typical for SE research by Garousi et al. (2019) and with the type “pre-prints” by the author of this thesis. The decision to classify pre-prints as Tier 1 was based on the fact that they cannot be considered white sources since no peer reviews have been completed. However, they still typically follow rigorous academic processes. The scoring was attained using a 3-point Likert scale (yes=1, partially=0.5, and no=0) for each question, and the overall score was normalized. There is no fixed threshold for exclusion in the scientific literature, as the quality control required depends on the research objectives. The threshold chosen for this work was 0.5 to capture a wide range of opinions from the industry.

[69] Cf. Wohlin (2014).

[70] https://ahrefs.com/de/backlink-checker

[71] See Tyndall (2010).



3rd Tier Grey Literature
Low Outlet Control / Low Credibility

Blogs, Emails, Tweets

2nd Tier Grey Literature
Moderate Outlet Control / Moderate Credibility

Annual Reports, News Articles, Presentations, Videos, Q&A Sites (Such as Stackoverflow), Wiki Articles

1st Tier Grey Literature
High Outlet Control / High Credibility

Books, Magazines, Government Reports, White Papers, Pre-Prints

Figure 8: Shades of Grey Literature Modified From Source Adams et al. (2017)[72]

A total of seventy-four sources were included to obtain the results of this research. Among them, sixty-one made contributions to RQ 1 and forty-two to RQ 2. The appendix contains a graphical mapping of the source counts and RQs in Figure 22. In addition, the breakdown of the number of included sources per quality levels based on outlet types is shown in Figure 9. The share of gray literature included in this MLR is around twenty-three percent, with the majority belonging to the first tier.

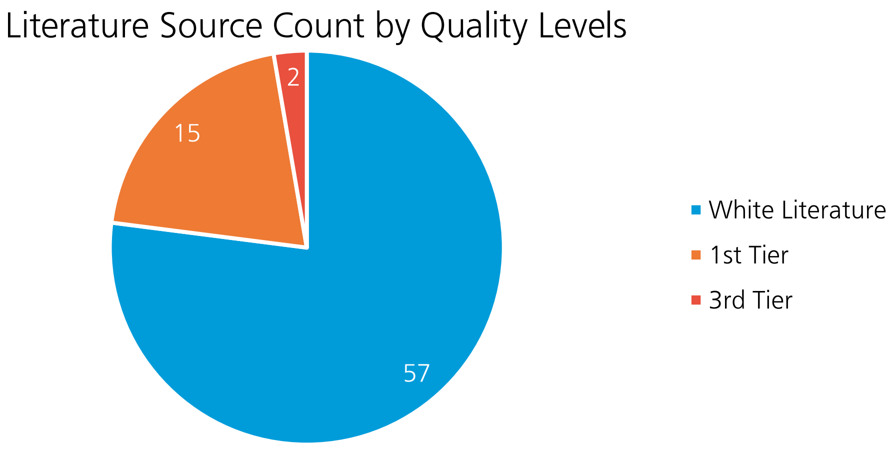


Figure 9: Literature Source Count by Quality Levels[73]

[72] Supplemented by Garousi et al. (2019) with media types relevant to software engineering and by the author of this thesis with the type "Pre-Prints".

[73] The final source pool did not contain any 2nd tier sources.

3.1.3 Data Extraction and Synthesis

For the data extraction, a spreadsheet was developed that combines quality assessment with a mapping of the literature to their contributions for each RQ, as Garousi et al. (2019) suggested.

The extracted data was first structured using the CRISP-DM Process stages, as shown in Figure 3. Then, this initial coding was continuously expanded if a finer granularity seemed appropriate or existing codes did not cover aspects. This approach corresponds to deductive coding, as Miles et al. (2014) described.

The MLR resulted in seventeen codes, i.e., challenges, for RQ 1. Figure 10 shows a mapping between source counts and challenges, while Table 17 in the appendix lists the respective sources. For RQ 2, the MLR resulted in fourteen codes, i.e., best practices. For these, the graphical representation is shown in Figure 11 and the listing of the corresponding sources in Table 18.

3.2 Focus Group

Focus groups are defined by Morgan (1996) as a "… research technique that collects data through group interaction on a topic determined by the researcher". The author deliberately kept this definition broad but included three elementary boundaries. First, the main objective of focus groups is to collect data for research purposes. Second, group interaction is the source of data. Third, the researcher determines the topic of interaction. Focus groups have their origin in commercial market research, applied in the 1940s by researchers at Columbia University regarding audience responses to radio soap operas.[74]

Since its initial publications, the method has become widely adopted in various research areas such as education, sociology, media and communications, and e-commerce.[75] In the Information Systems (IS) domain, focus groups were initially mainly used in the context of requirement engineering.[76] However, various voices from the academic field advocate for the broader use of focus groups in IS research, particularly regarding design science.[77]

Focus groups are beneficial when the topic under investigation is complex and challenging to quantify.[78] They are well suited to get a wide range of opinions, perceptions, and ideas on a given topic and help understand the perspectives of diverse backgrounds.[79]

Already in the early publications on focus groups, a distinction was made between the two use cases of hypothesis generation on the one hand and the interpretation of previously ascertained experimental findings on the other.[80] Similarly, today there are use cases like requirements engineering where focus groups are mainly used for idea generation as a more structured alternative to brainstorming sessions.[81] While in design science, focus groups are used to validate existing design artifacts.[82] However, both use cases are characterized by a substantial explorative element since the validation not only aims to identify problems but also to derive potential improvements. In a publication specifically on focus groups in the context of design science, Tremblay et al. (2010) further distinguish between Exploratory Focus Groups (EFG)

[74] Cf. Bloor et al. (2002).

[75] E.g., Aziz (2015); Gilflores and Alonso (1995); Lunt and Livingstone (1996); Morgan (1996).

[76] E.g., Alexander and Beus-Dukic (2009); Lauesen (2002).

[77] E.g., Gibson and Arnott (2007); Sutton and Arnold (2013).

[78] Cf. Powell and Single (1996).

[79] Cf. Krueger and Casey (2015).

[80] See Merton and Kendall (1946).

[81] Cf. Lauesen (2002).

[82] See Gibson and Arnott (2007); Tremblay et al. (2010).

and Confirmatory Focus Groups (CFG). EFGs aim to achieve incremental improvements for the design artifact, while CFGs aim to demonstrate its usefulness in the field setting.

The design artifact developed for this work includes various components that interact with each other and require collaboration among teams from different business units. The topic is therefore complex and requires collaboration between different stakeholders, making the focus group well suited for validation. In the following, the detailed setup of the focus group conducted for this thesis is described.

3.2.1 Participants and Recruitment

Focus groups should include experts from different perspectives, i.e., people who understand the problem domain, who use the solution, and who develop, deliver, and maintain the solution.[83] The group needs to be sufficiently diverse to encourage discussions, but groups that are too heterogenous can lead to conflicts that suppress the opinions of individuals.[84] Participants should share similar socio-demographic characteristics such as sex, age range, ethnicity, and social background and feel comfortable talking with the moderator and each other.[85]

Many publications recommend that group participants should not know each other beforehand.[86] On the other hand, Kitzinger (1994) argues that pre-existing groups were often surprised at how different their views were, which caused the participants to review and reconsider their positions critically.

The optimal number of participants is disputed in the literature and ranges from four to eighteen.[87] However, a minimum of six with a maximum of eight to ten participants is commonly recommended.[88] The group size should be chosen depending on the research objective, i.e., small groups are more suitable if a topic is to be discussed in great depth. In contrast, larger groups are preferable if various perspectives are examined.[89] Krueger and Casey (2015) consider large groups (eight participants or more) particularly suitable for piloting new

[83] Cf. Alexander and Beus-Dukic (2009).

[84] Cf. Bloor et al. (2002).

[85] Cf. Krueger and Casey (2015); Powell and Single (1996); Richardson and Rabiee (2001).

[86] E.g., Bloor et al. (2002); Krueger and Casey (2015); Powell and Single (1996).

[87] See Kitzinger (1994); Lauesen (2002).

[88] E.g., Bloor et al. (2002); Powell and Single (1996); Rabiee (2004).

[89] Cf. Krueger and Casey (2015); Morgan (1996).

ideas or materials. A widespread problem of focus groups is the non-attendance of participants. Therefore, an over-recruiting of ten to twenty-five percent is often recommended.[90]

The focus group session conducted for this work was performed in collaboration with the AI department of a medium-sized German software manufacturer. The roles of Product Lead Platform, Product Owner, Data Scientist, Platform Environment Engineer, Development Lead, and Software Engineer were represented in the focus group. A detailed list of the participants and their professional backgrounds can be found in Table 19 in the appendix. The first two roles can be associated with the understanding of the problem domain, as they must constantly deliver reliable new machine learning capabilities in their interactions with customers. The Data Scientists are the primary users of the implemented design artifact in their effort to develop new machine learning models. The Platform Environment Engineer represents the implementation and maintenance perspective, while the Software Engineer would use the models served by the MLOps solution by integrating them into systems.

Through this diverse group constellation, this thesis aimed to achieve a broad validation that ensures the design artifact's general utility, quality, and efficacy. Accordingly, a more detailed examination of the individual components was not the scope of this validation. The danger of conflicts that can lead to the suppression of individual opinions in diverse group constellations was known to the moderator and could have been counteracted appropriately.

Since the participants were employees of the same company, they knew each other in advance. Although this is a deviation from most of the literature recommendations on focus groups, the design artifact within this thesis requires collaboration between different departments of a specific organization. If employees from different companies would have been recruited for the focus group, there would be a severe risk that the department structures and processes between the companies would diverge to such an extent that the validity of the holistic design could no longer be ensured.

The sociodemographic characteristics were somewhat homogeneous within the group regarding social background, while there was limited homogeneity regarding age, ethnicities, and sex. However, since the homogeneity of sociodemographic characteristics is mainly about openness in the discussion, the existing familiarity between the participants might compensate here.

[90] Cf. Powell and Single (1996); Rabiee (2004).

Eight of the nine invited guests took part in the focus group. All invited roles were still represented, as only one product owner did not participate. Since the literature recommendation regarding over-recruitment was applied, the number of participants was in line with the expectations.

3.2.2 Number and Duration of Sessions

There is no universal recommendation on how many sessions are required; Powell and Single (1996) recommend one to ten, Krueger and Casey (2015) three to five, and Morgan (1996) four to six. Lauesen (2002) also outlines that multiple sessions are often useful, but if resources are limited, even one or two sessions will be helpful. The most valuable guidance is saturation, i.e., conducting focus groups until no new relevant information emerges.[91]

The duration of focus groups is less controversial and is often recommended at one and a half to two hours.[92] However, Kitzinger (1995) states that the sessions can sometimes take an entire afternoon.

For the research in this work, a single session of two hours was scheduled due to limited time resources. While certainly information saturation was not reached with one session, it has made a valuable contribution by highlighting key improvements.

3.2.3 Meeting Setting and Procedure

The location of the meeting should be neutral, i.e., without special significance for the participants and unrelated to the topic of the study.[93] Participants ideally should be arranged in a U-shape or circle to maximize eye contact.[94] Snacks and refreshments during the meeting are encouraged, as eating together promotes conversation.[95]

The vital role of the moderator is to facilitate open and uninhibited exchange between participants.[96] To achieve this, the moderator should be relaxed, non-judgmental, and a good listener, ideally sharing characteristics such as age, gender, and language. In addition, the

[91] Cf. Krueger and Casey (2015); Lauesen (2002).

[92] Cf. Krueger and Casey (2015); Morgan (1996); Powell and Single (1996).

[93] Cf. Powell and Single (1996).

[94] Cf. Krueger and Casey (2015); Powell and Single (1996).

[95] Cf. Krueger and Casey (2015).

[96] Cf. Powell and Single (1996).

moderator must understand the purpose and the topic of the study so that he or she can answer substantive questions in detail.[97] Finally, in design science research, the moderator is often one of the designers of the artifact; here, it is crucial to take care not to introduce any personal bias in the presentation of the artifact.[98] There are also approaches where multiple moderators, additional note-takers, or a guardian to ensure the neutrality of the moderator are involved in the focus group.[99]

Nowadays, virtual focus groups are also an option, which is often less expensive, more flexible in terms of time, and can be conducted with a larger target group due to geographical independence.[100] However, as with all online discussion groups, one must expect some "lurkers" who do not actively participate in the discussion. There are no established guidelines for virtual focus groups yet, but a schedule strictly adhered to by the moderator helps improve participation.

A focus group for validating a design artifact should start with an explanation of where and how the artifact could be utilized, followed by a questioning route.[101] Artifact design materials should be distributed to the participants at least one week in advance, so they have enough time to prepare.[102] The questioning route determines the agenda of the focus group and is directly aligned with the research objectives. A two-hour focus group should be composed of no more than ten to twelve questions, according to Krueger and Casey (2015). Powell and Single (1996) suggest using only five to six guiding questions, which the moderator should follow up with sub-questions. The questions must be open-ended, clearly formulated, and easy to understand.[103]

For evaluation purposes, the discussion is often recorded on video or audio but may also be tracked using notes, flipcharts, computer-based rapid transcripts, or memory logs.[104] Krueger and Casey (2015) state that video recordings are rarely worth the additional effort of audio recordings because facial expressions are barely visible through most wide-angle cameras in

[97] Cf. Krueger and Casey (2015); Tremblay et al. (2010).

[98] Cf. Tremblay et al. (2010).

[99] Cf. Krueger and Casey (2015); Powell and Single (1996); Tremblay et al. (2010).

[100] Cf. Krueger and Casey (2015).

[101] Cf. Tremblay et al. (2010).

[102] Cf. Gibson and Arnott (2007).

[103] Cf. Powell and Single (1996).

[104] Cf. Krueger and Casey (2015).

conference rooms. Participants must be informed of the recording before it begins, and most institutional review boards require written consent from all participants.[105]

The hosted focus group aimed primarily at identifying incremental improvements to the developed design artifact and therefore classified as an EFG. It was held in a conference room of the company whose employees attended to avoid causing additional travel time. However, some participants attended via online conference. The conference room used is designed for such hybrid meetings so that it is equipped with the necessary audio and video technology and a large screen to minimize the disadvantages of spatial separation. Refreshments and coffee were provided for on-site participants.

The design artifact was distributed to all participants one week in advance. At the same time, all participants were comprehensively informed about the study procedure, the recording, and associated privacy considerations. The written informed consent was obtained from all participants using an adapted version of the sample informed consent form of the ethics committee of the Technical University of Darmstadt.[106] The author of this thesis moderated the focus group and was aware of the bias risk through extensive preparation and actively sought to keep the explanations as neutral as possible.

The moderator opened the focus group session by introducing the agenda as well as the goals and limitations of the session. There was no need for a round of introductions, as all participants and the moderator already knew each other in advance. Participants were asked about their general requirements for an MLOps solution in the first interactive part without direct reference to the communicated design. This aimed to uncover potential requirements not addressed by any of the components represented in the design artifact. Afterward, the moderator explained the design in general and all components in detail and replied to the participants' questions.

The guiding questions of the session began with a question about the overall feasibility of the design. Additionally, a question about missing components was asked to ensure that the design represents a holistic perspective. This was followed by two questions focused on the component's importance. Subsequently, some component-specific questions were asked to identify weaknesses in individual components. Afterward, the participants were asked if they would like to discuss a particular aspect in more detail. Finally, an open question was asked to allow participants to discuss aspects that were not covered by the previous questions.

[105] Cf. Tremblay et al. (2010).

[106] See Hartmann (2021).

The questionnaire was structured with a focus on moving from an inclusive perspective to more specific details. In addition, the questions were ordered according to their priority to have the option to discuss important aspects in more depth by reducing the time for the following questions. All twelve guiding questions can be found in Table 20 in the appendix.

3.2.4 Data Extraction and Synthesis

The analysis must be practical, systematic, verifiable, and aligned with the study's purpose.[107] Practical here implies that a trade-off must be made between the resources available and the value of a more in-depth analysis. Systematic means following a prescribed sequential process that is deliberate and planned. Verifiable refers to the ability of another researcher to use the same data and arrive at similar findings.

The approach to analysis is the most controversial part related to focus groups in research, and there is no universally accepted technique.[108] Moreover, analysis methods used in other qualitative research methods cannot simply be applied, as there are some crucial differences with corresponding implications for focus groups.[109] For example, the use of words varies from individual to individual, making analysis methods based on counts in conversations more difficult than in speeches or individual interviews. Further differences can be found in Table 21: Features of Conversations by Krueger and Casey (2015) in the appendix.

A common approach for the analysis is coding based on a detailed transcript of the sessions.[110] This involves reviewing the transcript several times and assigning codes and conceptual categories, with the guiding questions serving as initial categories.[111] However, Krueger and Casey (2015) state that full transcripts are not always necessary; abbreviated transcripts, transcription of individual quotes, or analysis based on video/audio recordings, memory, and field notes are also possible. The main disadvantage of transcript-based analyses is the high amount of time and effort required for transcription, which can take between four and twelve hours for a two-hour focus group.

[107] Cf. Krueger and Casey (2015).

[108] Cf. Carey (1995).

[109] Cf. Krueger and Casey (2015).

[110] See, e.g., Gibson and Arnott (2007); Powell and Single (1996); Tremblay et al. (2010).

[111] Cf. Powell and Single (1996).

For coding and classification, Krueger and Casey (2015) recommend applying the following process based on four questions to each quote:

1. *Did the participant answer the question that was asked?*
    a. *IF YES → Go to question 3.*
    b. *DON'T KNOW → Set it aside and review it later.*
    c. *NO → Go to question 2.*
2. *Does the comment answer a different question in the focus group?*
    a. *IF YES → Move it to that question.*
    b. *IF NO → Put in discard pile (but don't throw the discard pile away until you are done with analysis).*
3. *Does the comment say something of importance about the topic?*
    a. *IF YES → Tape it to the newsprint under the appropriate question. (Or if you are working on a horizontal surface, just start a pile.)*
    b. *IF NO → Put in discard pile.*
4. *Is it like something that has been said earlier?*
    a. *IF YES → Start grouping like quotes together. Basically, you are making piles (categories) of like things.*
    b. *IF NO → Start a separate pile.*

Once all data is coded and classified, the codes and categories must be prioritized for the analysis. The comparison strategies applied here are based on frequency, extensiveness, intensity, specificity, internal consistency, and perceived importance by the participants.[112] Frequency refers to *how often* a concept was mentioned, extensiveness to *how many different people* mentioned the concept, and intensity to *how much the participant expressed passion or force*. Specificity refers to the *level of detail* provided by the participant, internal consistency to the *consistency of a participant's comments* during the session, and perceived importance to the *explicit expression of importance* by a participant. The prioritized codes then form the basis for the written report of the results.

The focus group session conducted for this thesis was recorded as a video for analysis. The decision to record with video was made because the technical equipment of the conference room allowed recording as video without any additional effort compared to an audio-only recording. However, in the analysis for this thesis, there was little additional value to be gained from a single wide-angle camera over audio-only recordings, as Krueger and Casey (2015)

[112] Cf. Krueger and Casey (2015).

already pointed out. Furthermore, since all participants knew each other beforehand and were used to video recordings from their everyday work, no adverse side effects are to be expected from the awareness of the video recording.

The focus group was scheduled to last two hours but was extended by a few minutes in response to the request from some participants to provide time for more in-depth discussions. Unfortunately, due to the spontaneity of the extension, two of the Data Scientists and the Product Lead Platform could only attend for the scheduled two hours. Nevertheless, especially since the topics discussed in the additional minutes were more in-depth technical details, I consider this to be an added value to this work. Furthermore, I assessed the impact of a distorted impression due to the missing participants in the last minutes as low since it was only a matter of technical details. Therefore, I decided that the value of the additional information is more significant and included the last minutes in the analysis. The total length of the recording is therefore two hours and twenty-three minutes.

The entire video recording was fully transcribed for analysis, resulting in a thirty-six-page (approximately 19,000-words) transcript. For the subsequent coding and classification, the literature recommendations were applied by first adopting the guiding questions as categories. The procedure was based on the four process questions, although a wide spreadsheet was used for the grouping instead of paper-based piles. When analyzing a single session using codes, there is a risk that essential topics will not be mentioned frequently due to time constraints.[113] Therefore, extensiveness, intensity, specificity, and perceived importance were weighted higher than frequency to counteract this.

[113] See Krueger and Casey (2015).

4 Challenges due to the Adoption of Machine Learning

The challenges that organizations face in the adoption of machine learning in production systems arise in various areas. They go beyond those of traditional software development, as changes in machine learning systems no longer result only from code changes but also from changes in the data or model.[114] Consequently, there is increased complexity in the operation of these applications and the infrastructure required for their development. In addition to development and deployment, some challenges arise from regulations or the involvement of other stakeholders.

Figure 10 provides an overview of the challenges identified based on the results of the MLR conducted for this thesis. The bar length represents the number of sources that have emphasized the specific challenge, and a list of sources that contributed to the counts can be found in Table 17 in the appendix. In the following subsections, the challenges are examined in more detail. The deviation in the order between the subsequent explanations and the graphical representation is intended to facilitate understanding of the interrelationships between the challenges.

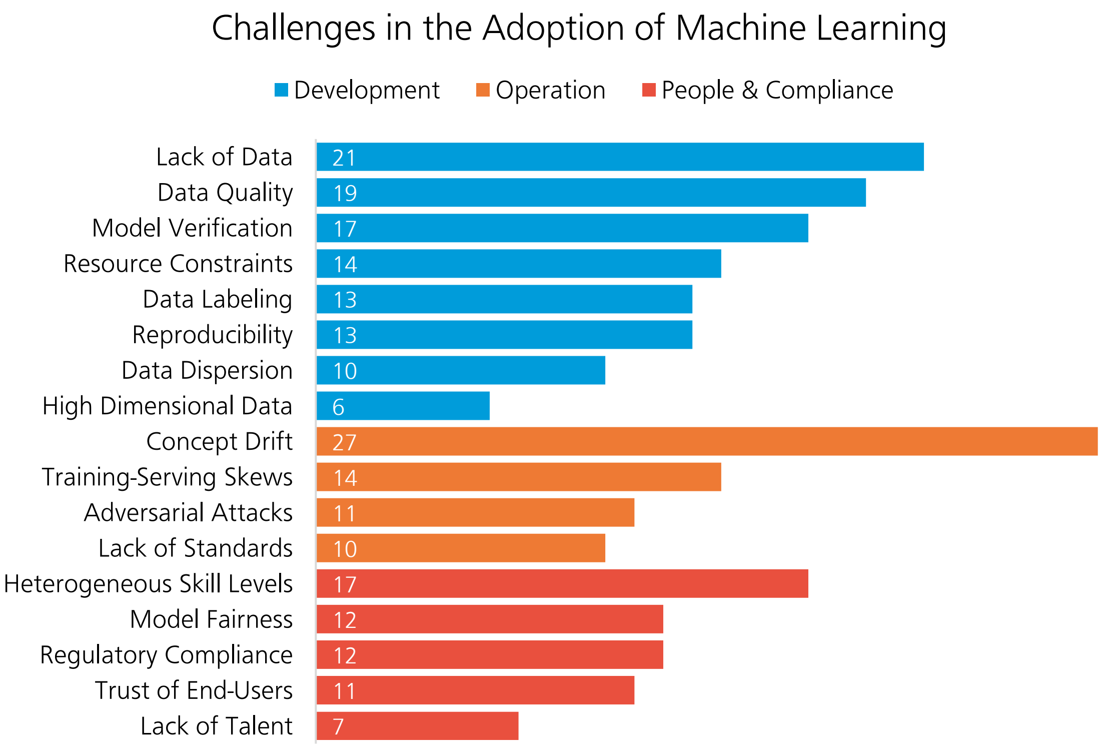


Figure 10: Challenges in the Adoption of Machine Learning

[114] Cf. Paleyes et al. (2020).

This chapter first discusses the challenges associated with the *development* of machine learning systems. It then examines the challenges that can arise during *operation* and finally addresses challenges related to *people & compliance*. However, it is vital to view organizations as a cohesive system, as unsolved challenges in individual areas can negatively impact all other areas.

4.1 Development

Data Availability: The core of the development of machine learning models is data. Therefore it is not surprising that data availability is a significant challenge.[115] Besides identifying data sources, this also involves applying enrichment techniques to extend the present data.[116] Enrichment techniques augment existing data with external information, e.g., a date could be augmented with information about whether it is a public holiday.[117] Another way to enrich data is to use different transformations of existing data, such as applying different embedding techniques to text data.[118] Polyzotis et al. (2017) state that the key challenge here is identifying which transformations and augmentations can enrich the data in a meaningful way.

Data Quality: More data does not always lead to more accurate models if the data quality is not sufficiently high; therefore, data quality is another major challenge.[119] Two common issues are the detection and handling of errors in the data and potential biases introduced by a lack of variance.[120] The data validation to detect errors is concerned mainly with the detection of dirty data, e.g., duplicates with different spellings, changes due to evolutions of the data source, e.g., the unit of a feature could change from days to hours, and missing data. [121] While errors in the data are relatively easy to detect and handle, detecting biases and unexpected distributions of the data is often more challenging.[122] Moreover, unbalanced datasets are a common problem in many use cases due to the initial rarity of the events under consideration.[123]

[115] E.g., Amershi et al. (2019); Paleyes et al. (2020); Polyzotis et al. (2017).

[116] Cf. Polyzotis et al. (2017).

[117] Cf. Heuvel and Tamburri (2020).

[118] Cf. Polyzotis et al. (2017).

[119] E.g., Amershi et al. (2019); Stonebraker and Rezig (2019); Whang and Lee (2020).

[120] See Paleyes et al. (2020).

[121] Cf. Whang and Lee (2020).

[122] See Paleyes et al. (2020).

[123] Cf. Kocheturov et al. (2019).

Data Dispersion: The data used for model training often comes from different data sources and must be joined accordingly.[124] The associated challenges may arise from different schemas, different conventions, as well as different storage and access methods.[125]

Noisy or Missing Labels: Many machine learning systems today are based on supervised learning techniques and rely on labeled data. The availability of large amounts of high-quality labeled data is a common challenge in this context.[126] A reason for this is limited access to domain experts in cases where expertise is necessary to interpret and label the raw data appropriately.[127] The typical approaches for labeling are semi-supervised learning and crowdsourcing techniques, but there are also more sophisticated methods such as active learning.[128] In semi-supervised learning approaches, existing labels are used to predict the labels of unlabeled data as much as possible. Crowdsourcing is based on the manual labeling of many participants and is accordingly primarily suitable for data that does not require an elevated level of confidentiality or specific expert knowledge. The lack of expert knowledge of participants in crowdsourcing projects can be mitigated to some extent by multiple labeling by different participants for each record.[129] Active learning approaches specifically attempt to label only those records likely to improve the model accuracy.

High Dimensional Data: Nearly all the challenges described above are further complicated by the high number of features that many use cases bring with them. Analyzing data with many dimensions is an additional challenge in many practical applications of machine learning.[130] As the number of features increases, a more considerable number of records is inevitably required to ensure appropriate variance across different feature combinations. Furthermore, the data cleaning effort increases since more columns need to be cleaned. Additionally, the more considerable number of features offers more potential for different schemas and conventions, making merging multiple data sources more difficult. Consequently, aggregating all features to a suitable label is more difficult with an increasing number of features.

Resource Constraints: The training of models with substantial amounts of high-dimensional data places great demands on the hardware, which results in several challenges. The hardware

[124] E.g., Dong and Rekatsinas (2018); Paleyes et al. (2020); Salama et al. (2021).

[125] Cf. Paleyes et al. (2020).

[126] E.g., Baier et al. (2019b); Paleyes et al. (2020); Schelter et al. (2018).

[127] E.g., Baier et al. (2019b); Budd et al. (2021); Paleyes et al. (2020).

[128] Cf. Whang and Lee (2020).

[129] Cf. Budd et al. (2021).

[130] E.g., Amershi et al. (2019); Cai et al. (2018); Kocheturov et al. (2019).

infrastructure costs associated with machine learning are a significant problem, particularly for small and medium-sized enterprises.[131] However, even in cases where infrastructure costs are less important, large amounts of data can cause the training of models to become unacceptably time-consuming.[132] This becomes even more problematic in cases that require frequent retraining, where a high training latency can prevent the use case from succeeding.[133] In addition, high demands on hardware also cause increased energy consumption, resulting in a negative impact on the environment.[134]

Reproducibility: Technologies for the versioning of code have been established in software development for many years. However, the performance of machine learning models is highly dependent on the data used for training, for the versioning of which there are no established standards yet.[135] Accordingly, keeping track of experiments is a widespread challenge in the real-world development of machine learning systems.[136] To ensure that experiments are reproducible, both a version of the training code and a version of the input data must be kept and stored so that they can be linked to an experiment.[137]

Model Verification: Verifying the model performance is another obstacle that hinders organizations in using machine learning models in production systems.[138] One reason is that defining correctness criteria for each possible input scenario is difficult or impossible.[139] In addition, the utilized metrics are often business-driven and challenging to optimize.[140] For example, the goal of a frequently answered question application based on machine learning might be to avoid as many support requests as possible. However, the metric "avoided support requests" is difficult to capture for the optimization of the model. The situation is further complicated by the fact that in many production systems, multiple models interact with each other and thus cannot be considered in isolation.[141]

[131] See, e.g., Algorithmia (2021); Amershi et al. (2019); Heizmann et al. (2022).

[132] Cf. Zhang et al. (2019).

[133] Cf. Hazelwood et al. (2018).

[134] Cf. Paleyes et al. (2020).

[135] Cf. Amershi et al. (2019).

[136] E.g., Algorithmia (2021); Salama et al. (2021); Schelter et al. (2017).

[137] Cf. Zaharia et al. (2018).

[138] E.g., Amershi et al. (2019); Ishikawa and Yoshioka (2019); Paleyes et al. (2020).

[139] Cf. Ishikawa and Yoshioka (2019).

[140] Cf. Paleyes et al. (2020); Schelter et al. (2018).

[141] Cf. Amershi et al. (2019).

4.2 Operation

Training-Serving Skew: Despite rigorous validation, it often happens in practice that machine learning systems behave differently than expected in operational use.[142] The cause of such discrepancies are inconsistencies in terms of data or runtime dependencies between training and serving environments.[143] Concerning the data, it may be that the previously used data was not representative or that the input data is handled differently in the production system.[144] Real-world application data from production systems is often very skewed, so a few categories are widespread, and many are rare.[145] Another cause can be the integration in the production environment when the machine learning model is embedded in a more extensive system, and there is a feedback loop between the application code and the machine learning model.[146]

Concept Drift: A concept in the context of supervised learning is described by the joint distribution $P(X, Y)$, with input space X and target space Y.[147] Concept drift refers to any change in distribution and is a common operational challenge for machine learning systems, as data streams often change over time.[148] Furthermore, it is often assumed that even if the distribution changes, both feature space X and label space Y remain identical, which is not always the case in practice.[149] A change in X or $P(X)$ is referred to as data drift and results in a loss of representativeness of the training or evaluation data. Model or real drift refers to a change in $P(Y|X)$. Such change is often caused by changes in $P(X)$ but can also occur without.[150] Undetected and unaddressed drifts can lead to a decrease in prediction accuracy and introduce biases.[151]

Adversarial Attacks: Another challenge in the operation of machine learning systems are malicious attacks on the machine learning models.[152] For example, in the case of a machine learning system for intrusion detection, an adversarial attack could try to modify the training data generated from the live application in such a way that a later attack on other systems is

[142] E.g., Ishikawa and Yoshioka (2019); Salama et al. (2021); Schelter et al. (2018).

[143] Cf. Salama et al. (2021).

[144] Cf. Boutaba et al. (2018); Zinkevich (2021).

[145] Cf. Schelter et al. (2018).

[146] Cf. Zinkevich (2021).

[147] Cf. Gama et al. (2014).

[148] E.g., Klaise et al. (2020); Lu et al. (2018); Renggli et al. (2021).

[149] Cf. Renggli et al. (2021).

[150] Cf. Gama et al. (2014).

[151] Cf. Makinen et al. (2021).

[152] E.g., Algorithmia (2021); Balakrishnan et al. (2020); Salama et al. (2021).

not detected.[153] Such an attack through the training data is called data poisoning; other attacks are model stealing and model inversion.[154] Model stealing attacks primarily affect Machine Learning as a Service Provider (MLaaS), whose models can be used for a fee via APIs.[155] In such attacks unlabeled data is used to capture and replicate the knowledge of the model and thus the intellectual property of the MLaaS provider. In contrast to model stealing, model inversion attacks aim to discover the data used to train the model.[156]

Lack of Standards: Machine learning systems often involve multiple teams which use different languages, frameworks, and machine learning libraries, all of which need to be supported for operational use.[157] This diversity is because some technologies are better suited for specific subsets of problems and have various tradeoffs regarding flexibility, performance, and compatibility.[158] One exemplary related issue is that detecting semantic errors is much more difficult in an environment where many components must fit seamlessly as an overall architecture.[159]

4.3 People & Compliance

Heterogenous Skill Levels: Machine learning systems' development and operation often involve individuals and teams with vastly different qualifications. The various levels of expertise of the stakeholders often constitute a significant challenge for collaboration in corporate practice.[160] The stakeholders include the users of the machine learning systems, who may have unclear or unrealistic expectations about the model's capabilities.[161] On the other hand, it is not always easy for Data Scientists to accurately understand the needs of other stakeholders.[162]

Trust of End-Users: Another challenge closely related to the heterogeneous skill levels is gaining the trust of the end-users.[163] The explainability of model predictions is the most challenging

[153] Cf. Boutaba et al. (2018).

[154] Cf. Paleyes et al. (2020).

[155] Cf. Correia-Silva et al. (2018).

[156] Cf. Pal et al. (2019).

[157] E.g., Heuvel and Tamburri (2020); Paleyes et al. (2020); Zaharia et al. (2018).

[158] See Hazelwood et al. (2018).

[159] Cf. Martel et al. (2021).

[160] E.g., Amershi et al. (2019); Schelter et al. (2018); Zaharia et al. (2018).

[161] Cf. Makinen et al. (2021); Nahar et al. (2022).

[162] Cf. John et al. (2022).

[163] E.g., Heizmann et al. (2022); John et al. (2022); Klaise et al. (2020).

aspect in this context, as it is often even for data scientists difficult to understand them precisely.[164] In addition to confidence in model predictions, privacy concerns play an important role here, as user data is often used to further improve the machine learning systems.[165]

Model Fairness: Machine learning models influence the lives of many individuals with their predictions. One example of direct impact is the use of machine learning in human resource departments to select applicants.[166] However, there are also indirect influences, such as social media news feeds, which are often controlled by machine learning models and can affect the perceptions of many users.[167] Avoiding the discriminatory behavior of models in such use cases is often a significant challenge for companies.[168] The cause often resides in biased datasets, commonly generated by humans whose own biases are then reflected in the data.[169]

Regulatory Compliance: The privacy concerns and the threat of potential biases encouraged countries to impose increasing regulations on machine learning systems.[170] Regulators must always consider the essential tradeoff between the simultaneous need for personal information for machine learning systems to improve predictive quality in areas such as medical services while at the same time protecting the privacy of individuals.[171] Compliance with the multitude of new regulations is a significant challenge for companies developing and operating machine learning systems.[172]

Lack of Talent: Companies face another common challenge in acquiring the talent needed for machine learning applications.[173] This implies that Data Scientists are a scarce resource in the companies and must be carefully allocated to projects.[174] Beyond that, new challenges emerge related to the education and training of all internal stakeholders.[175]

[164] Cf. John et al. (2020).

[165] Cf. Chui et al. (2021).

[166] See Jati and Lhaksmana (2020).

[167] See Hazelwood et al. (2018).

[168] E.g., Paleyes et al. (2020); Salama et al. (2021); Whang and Lee (2020).

[169] Cf. Stonebraker and Rezig (2019).

[170] See, e.g., European Commisson (2020); The National New Generation Artificial Intelligence Governance Specialist Committee (2021); Weaver (2018).

[171] Cf. Malle et al. (2017).

[172] E.g., Chui et al. (2021); Makinen et al. (2021); Paleyes et al. (2020).

[173] E.g., Makinen et al. (2021); Paleyes et al. (2020); Thieullent et al. (2020).

[174] Cf. John et al. (2022).

[175] Cf. Amershi et al. (2019).

5 MLOps Best Practices and Open-Source Tools

MLOps has become increasingly popular in recent years due to the widespread adoption of machine learning systems in many organizations. However, it is still a very dynamic environment where no standards have been established yet, and new tools are launched regularly. Despite the novelty of the subject, the increasing popularity in academia and the cross-industry relevance in practical applications have already led to many publications and user reports from which valuable insights can be derived.

This chapter will start by outlining and explaining several best practices to support the establishment of a holistic MLOps architecture. All explanations refer to supervised learning use cases, as the labels constitute an additional level of complexity. Nevertheless, the best practices can largely be adapted to unsupervised learning environments and therefore provide valuable insights in this area as well.

After discussing the best practices, the chapter concludes with a description of various open-source tools that can serve for the practical implementation of MLOps capabilities in organizations.

5.1 Best Practices

The best practices described in the following sections were derived from the MLR conducted for this work. All resulting best practices are illustrated in Figure 11, where the bar length reflects the number of sources that support the respective practice. The detailed list of sources for each best practice can be found in Table 18 in the appendix. The following explanations are structured thematically and therefore differ in their order from the illustration in the chart, which uses color-coding to indicate in which subchapter the respective practice can be found.

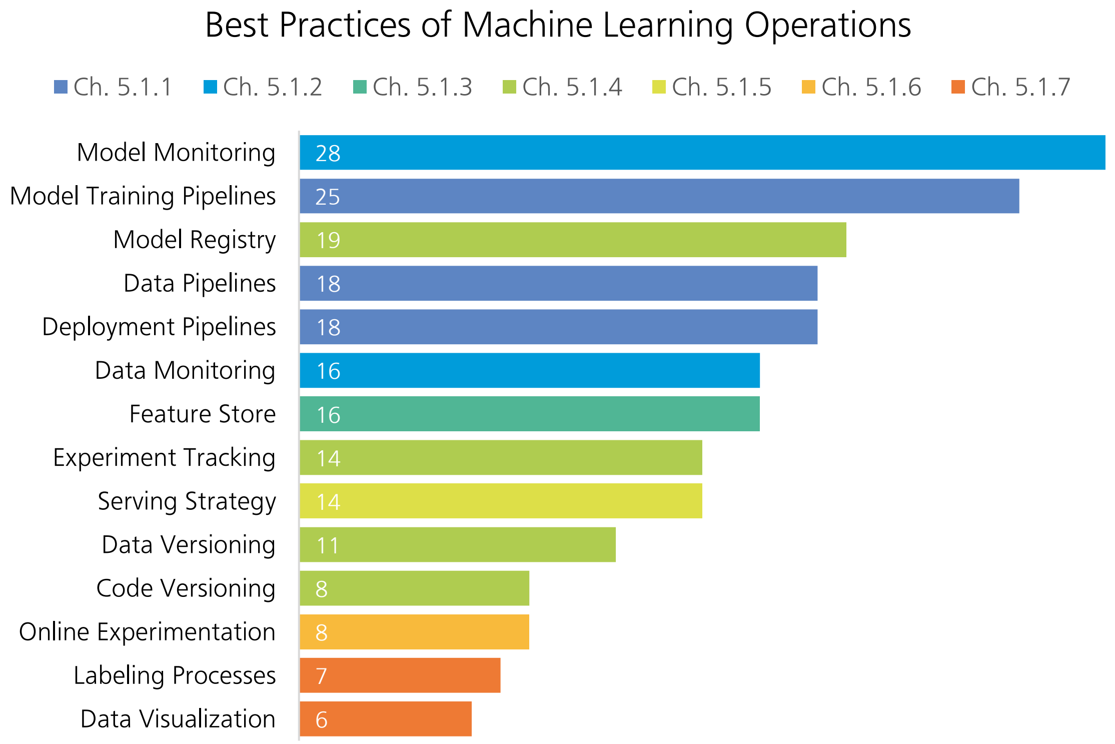


Figure 11: Best Practices of MLOps

5.1.1 Develop the Modeling, Not the Model

The traditional development of machine learning models, as defined in the CRISP-DM model (see Figure 3), relies on a static environment that at least approximately corresponds to what is represented in the training data. This assumption is usually not sustainable in the real world, where concept drift and training-induced biases are typical, implying that frequent modification of the models in production systems is indispensable. Unless the reusability of the operations performed during modeling has been considered from the beginning, the effort required to achieve this can be enormous. Therefore, MLOps aims to design pipeline operations that constitute reusable building blocks and minimize manual interactions. These are typically used by various pipelines that fulfill different objectives and interact with each other. Often data pipelines, model training pipelines, and deployment pipelines are necessary.[176] All pipelines should be designed in such a way that they can be version controlled, tested, and executed in the target environment.[177] It should also be possible to execute the pipelines locally for

[176] Cf. Symeonidis et al. (2022).

[177] Cf. Sato et al. (2019).

debugging and trigger them in production on demand, by a schedule, or by predefined events.[178]

Data Pipelines represent the process of extracting data from various sources, applying a series of transformation steps, and storing the output data in a database, file, or stream.[179] This process is widely used and is referred to as the ETL.[180] However, there are advocates for a clear separation between data ingestion and processing and favor an ELT process.[181] Regardless of the process sequence, providing data connectors and interfaces with appropriate encoders and decoders for a wide range of data sources is essential.[182]

Model Training Pipelines are also referred to as machine learning pipelines and combine data preprocessing, feature engineering, model selection, model optimization, and evaluation.[183] The preprocessing and feature engineering steps are like those of the manual machine learning development process, with the only difference being that they are designed as reusable building blocks. In the model selection stage, different machine learning algorithms are evaluated to select the most suitable one.[184] During the model optimization stage, the hyperparameters of the selected model are determined using automated search procedures before the final evaluation is performed. The exact definition of the pipeline depends on the particular use case and may therefore contain more or fewer steps. There should be a set of frequently needed predefined operations and the option to define new custom components.[185] For the initial pipeline, it makes sense to start with a minimal configuration and continuously expand it to facilitate potential debugging.[186] Beyond that, it is important to record information about the execution of the pipeline, the resulting model artifacts, and their metadata, such as evaluation results.[187] For each model in productive use, there should be clearly defined criteria for when retraining must occur.[188]

[178] Cf. Salama et al. (2021).

[179] Cf. Sato et al. (2019).

[180] E.g., Charrington (2020); Salama et al. (2021); Sato et al. (2019).

[181] See Tagliabue (2021).

[182] Cf. Salama et al. (2021).

[183] Cf. Sato et al. (2019).

[184] Cf. Granlund et al. (2021).

[185] Cf. Salama et al. (2021).

[186] Cf. Zinkevich (2021).

[187] Cf. O'Leary and Uchida (2020).

[188] Cf. Chui et al. (2021).

Deployment Pipelines in machine learning resemble CI/CD pipelines familiar from DevOps. They, therefore, include stages such as the packaging of models in a format suitable for production, testing, and deployment to different environments.[189] However, in machine learning, extending the established steps with steps such as model bias detection, fairness checks, or methods to improve explainability is recommended.[190]

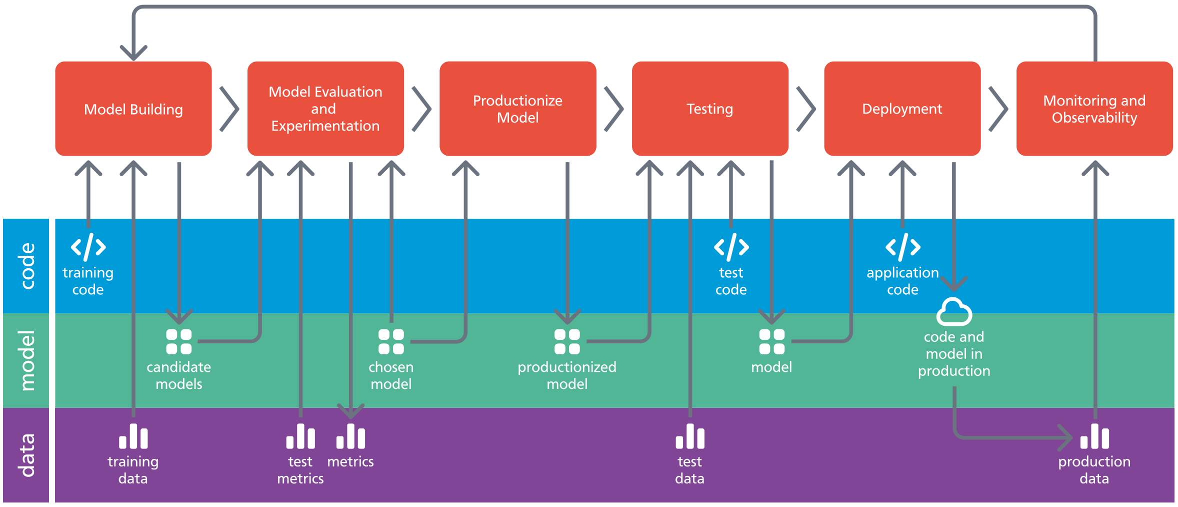


Figure 12: Machine Learning End-To-End Process by Sato et al. (2019)

All pipelines should be designed in such a way that they can be connected to form an end-to-end process.[191] Figure 12 illustrates the MLOps process as a combination of the model training and the deployment Pipeline. The first two stages of the process correspond to the model training pipeline, while the three stages from “Productionize Model” to “Deployment” match the deployment pipeline. The “Monitoring and Observability” stage is not an explicit component of the pipelines described above. However, it is an integral part of MLOps as new training data can be derived from it. Furthermore, the data pipelines are not explicitly represented, which is supposedly since they are associated with DataOps.

Pipelines directly address the challenges of *Reproducibility*, *Data Dispersion*, and *Lack of Talent*. The concept of pipelines already implies reproducibility. It is supported by appropriate logging of pipeline executions, while the challenge of data dispersion is mitigated by reusable components for data pipelines such as data connectors and transformation steps. Additionally,

[189] Cf. Granlund et al. (2021).

[190] Cf. Sato et al. (2019).

[191] See, e.g., Amershi et al. (2019); Karamitsos et al. (2020); Sato et al. (2019).

the reusability of components and the minimal effort required to re-run pipelines reduces the workload for the people involved and thus alleviates the challenge of talent shortages. Due to pre-defined workflows, several indirect facilitations can be identified, such as the simplified treatment of *Data Quality* defects, *Concept Drifts*, and *Training-Serving Skews*. Furthermore, the *Lack of Standards* for machine learning frameworks is not resolved, but reusable components facilitate the handling. Building *Trust of End-Users*, *Ethical Challenges*, and *Regulatory Compliance* can similarly be simplified with appropriate components to provide documentation and control fairness criteria.

5.1.2 Continuously Monitor the Input Data and the Model

In traditional software development, processes are determined solely by the code and, therefore, can be tested in detail. The program outputs usually depend on the inputs, but the processing steps are defined in advance. Apart from potential flaws in the code, internally, only the utilization of the systems varies, so monitoring primarily covers the infrastructure (e.g., CPU/GPU/Memory/Storage utilization, throughput, latency).

Machine learning models adhere to the patterns discovered during the model's training and are therefore similarly static, but the internal flow of information is largely unknown ("black box" character).[192] Therefore, the verification of such models is usually based on statistical approaches with many test examples, which must be representative of the cases in the production environment to provide meaningful results. However, the input data faced by the machine learning systems in production environments are often unknown in advance and may change over time. To ensure the reliable operation of machine learning systems, it is therefore essential to carefully monitor the incoming data and the model, including the predictions and regular infrastructure monitoring. When anomalies are detected, actionable alerts should be triggered and a dashboard available that summarizes all relevant information.[193]

Data Monitoring seeks to detect changes between the current incoming data and the data used for training or historical incoming data. It is crucial as it determines previous evaluations' predictive quality and validity.[194] Table 1 provides an overview of some critical indicators that should be observed in data monitoring. Once anomalies are identified, it is crucial to analyze

[192] Cf. Heizmann et al. (2022).

[193] Cf. Zinkevich (2021).

[194] Cf. Schelter et al. (2018).

how severe their impact is and what caused them. The causes of data anomalies may be bugs in the code that collects or transforms the data, miscommunications between teams working on different parts of the pipeline, or production hiccups such as using binaries with inconsistent versions.[195] However, the cause may also be a severe change in the incoming data, which requires retraining the model with an updated dataset or even remodeling.[196]

Table 1: Data Monitoring Indicators Inspired by Ruf et al. (2021)

Indicator	Description
Data Schema Validity	The validity of the data schema is given if exactly the features used in the training data are also present in the serving case. It is an essential measure of data integrity and ensures a minimum level of data quality by demanding completeness.[197]
Data Consistency	The data consistency metric also aims to ensure data quality. Potential criteria here are whether the values are in the known value ranges, if the data types of the features correspond to the training data, and whether there are errors in the format.[198]
Outliers	An observation that deviates from other observations to such an extent that it is suspected to have arisen by some other mechanism is termed an outlier.[199] Outliers are a type of data inconsistency but should be examined closely as additional insights may be derived from them.[200] Since many algorithms are sensitive to outliers, such cases should be flagged because the results are unreliable.[201]
Data Drift	A change in the statistical properties of the input data, e.g., a change in its distribution, is referred to as data drift.[202] Such shifts reduce the representativeness of the training data and may indicate the need for retraining.
Bias / Fairness	To avoid biases in the predictions, it is vital to verify the balance of the incoming data when it is used for model training. Particular attention should be paid to features that are known to induce biases, such as race, gender, age, and income groups.[203]
Training-Serving Skews	The deviation between serving data from production and training data is referred to as training-serving skew.[204] Such deviations imply that the training data is not representative of real-world use, which means that the validity of the predictions is not ensured.

[195] Cf. Polyzotis et al. (2017).

[196] Cf. Schelter et al. (2018).

[197] Cf. Ruf et al. (2021).

[198] Cf. Ruf et al. (2021).

[199] Cf. Hawkins (1980).

[200] Cf. Ruf et al. (2021).

[201] Cf. Klaise et al. (2020).

[202] Cf. Ruf et al. (2021).

[203] Cf. Sato et al. (2019).

[204] Cf. Polyzotis et al. (2017).

Model Monitoring aims to monitor both resource utilization and model predictions. The metrics for resource utilization are similar to those in traditional application development (e.g., CPU/GPU/Memory/Storage utilization, throughput, latency) and thus evaluate the model's efficiency.[205] This is especially important when the available resources are limited (e.g., the model is deployed on mobile devices), or the infrastructure cost is critical to the business case. Table 2 highlights indicators that should be considered in model monitoring. The precise definition of the metrics with their threshold values for alerting depends on the individual case.

Continuous Monitoring thus directly addresses the challenges of *Training-Serving Skews*, *Concept Drift,* and *Adversarial Attacks* by identifying anomalies at an early stage. Furthermore, the early detection of biases contributes to their reduction in future model versions resulting in an improvement in *Model Fairness*. Finally, proactively responding to biases and defects may also increase the *Trust of End-Users*. Therefore, monitoring the entire machine learning system is a core element for mature, automated, robust, and efficient MLOps systems.[206]

[205] See Salama et al. (2021).

[206] Cf. Symeonidis et al. (2022).

Table 2: Model Monitoring Indicators Inspired by Ruf et al. (2021)

Indicator	Description
Output Distribution	The disparity between the prediction distributions in training and serving indicates model degradation.[207]
Predictive Performance	The definition of predictive performance metrics requires domain knowledge, and their calculations often rely on labels.[208] Typical metrics include error rates, accuracy, Area Under the Curve (AUC), Receiver Operating Characteristics (ROC), precision, and recall.[209] It is crucial to have a clear and consistent definition of the metrics to ensure comparability over time.[210]
Feature Importance Change	Feature importance describes the contribution of each feature to the prediction. A change in the importance of features may indicate a drift is occurring.[211]
Concept Drift	Concept drift refers to a situation in which the patterns learned by the model are no longer valid.
Bias / Fairness	The model may have learned biases from the training data that discriminate against certain groups. Therefore, in some situations, it may be helpful to set pre-defined rules that override the model's prediction to avoid future bias.[212]
Numerical Stability	The model may have learned invalid or incorrect numbers in training, which do not trigger an error but result in significantly erroneous predictions.[213]

5.1.3 Establish a Centralized Location for Storing and Documenting Datasets

The development of machine learning models is fundamentally based on the data used to identify patterns. The effort associated with data collection and processing is difficult to quantify in general terms and depends on the use case. However, it is evident in the literature that it accounts for a significant portion of practitioners' workload.[214] Some voices even state that this constitutes ninety percent of their working time.[215] Therefore, the availability of sufficient and high-quality data is, in many cases, a decisive success criterion for machine

[207] Cf. Ruf et al. (2021).

[208] Cf. Ruf et al. (2021).

[209] Cf. Sato et al. (2019).

[210] Cf. Schelter et al. (2018).

[211] Cf. Ruf et al. (2021).

[212] Cf. Sato et al. (2019).

[213] Cf. Ruf et al. (2021).

[214] See, e.g., Amershi et al. (2019); Baier et al. (2019a); Polyzotis et al. (2017).

[215] Cf. Stonebraker and Rezig (2019).

learning systems.[216] The most significant marginal gain is almost always in the data; making clean data accessible is significantly more important than minor model improvements.[217]

Feature Stores provide a centralized location for storing features, datasets, and their documentation within the organization, thereby bridging the gap between raw data sources and model training as well as serving.[218] Strict data quality policies should be in place to ensure that only trustworthy and high-quality data is present in the feature store with synthetic data generated when sufficient natural data is not available.[219]

Feature stores can facilitate shareability between teams and the discovery and reusability of existing datasets.[220] Such a central store can be a good starting point for new machine learning teams and can help enhance existing models with new features.[221] Centralized storage of features and datasets counteracts the issue of *Data Dispersion*, while improved discoverability mitigates the challenge of *Lack of Data*. Furthermore, having clearly defined quality criteria in place and building on what other teams have already accomplished also mitigates the problem of *Data Quality*. Centralization also allows greater control over *Regulatory Compliance*, while reusability reduces the amount of work required and thus counteracts the *Lack of Talent*.

5.1.4 Ensure Reproducibility by Versioning Code, Data, and Models

Using version control systems such as git[222] has become indispensable in application development. However, the established practices can only be applied to machine learning system development to a minimal extent due to interdependencies with data and trained models. The data is often stored in systems for which there are no established versioning standards.[223] Additionally complicating the problem is that machine learning models often emerge from many experiments conducted in computational notebooks. Computational notebooks are systems that use text-based programming languages (e.g., Python or JavaScript) and interweave them with explanatory text and the program outputs in a single document.[224]

[216] See Hazelwood et al. (2018).

[217] Cf. Tagliabue (2021).

[218] Cf. Lakshmanan et al. (2020).

[219] See Balakrishnan et al. (2020).

[220] Cf. Salama et al. (2021).

[221] Cf. Hazelwood et al. (2018).

[222] https://git-scm.com

[223] Cf. Amershi et al. (2019).

[224] Cf. Lau et al. (2020).

Code Versioning requires additional attention when MLOps practices are introduced, despite the well-established standards in traditional application development. The use of versioning integrations for computational notebooks is one option.[225] An extensive comparison of the different versioning systems for computational notebooks has been compiled by Lau et al. (2020). However, for use in production, scripts are still recommended; while computational notebooks have some advantages for experimentation, they do not outweigh the benefits of regular scripts.[226] For example, notebooks complicate the reuse of functions since they usually consist of only one large file and are therefore less modular. Furthermore, notebook files are inevitably more cluttered than regular scripts since they contain outputs and descriptive texts.

Data Versioning is essential to ensure the reproducibility of experiments and models. However, data versioning is more complex than code versioning due to the substantial number of different data formats. The two main approaches to data versioning are git-style and time-difference-based approaches.[227] The git-style approaches store immutable versions of the data and assign them to individual commits with appropriate metadata. This has the advantage of being file-format agnostic and easy to integrate into existing git workflows. However, they also lead to copies of similar records that need to be stored, resulting in vast amounts of data. On the other hand, the time-difference-based approaches store only the modifications to the last version. They are correspondingly more economical in terms of storage requirements but not agnostic to file formats and not as easy to integrate into existing git workflows.

Model Registry refers to a system for versioning model artifacts with all relevant metadata. It is, therefore, a specific case of data versioning intended to facilitate accessibility to models with their variety of metadata. The metadata includes all necessary information to reproduce the model, such as the model architecture, which training dataset was used, and the applied hyperparameters.[228] Additionally, documentation about the model should be stored containing information about known issues and tradeoffs, the model's usage, and runtime dependencies.[229] Furthermore, critical data for model management such as evaluation results, quality metrics used for validation, and the deployment status should be recorded.[230]

[225] Cf. Salama et al. (2021).

[226] Cf. Goes (2021).

[227] Cf. Ruf et al. (2021).

[228] See Schelter et al. (2017).

[229] Cf. Chui et al. (2021); Salama et al. (2021).

[230] See Ruf et al. (2021); Salama et al. (2021).

Experiment Tracking is enabled by code and data versioning and is often an integral part of model registry systems. It aims to rigorously record all information necessary to reproduce an experiment and its evaluation results to ensure comparability.[231] This includes information about the training data, the hyperparameters, the evaluation metrics, and metadata about the dataset used for evaluation. Furthermore, it is the foundation for deciding which models should be promoted to production. Therefore, it is essential to have easy-to-use integrations for both the local experiments and the pipeline ecosystem. In addition, there should be simple and meaningful visualization options for comparing experiments.[232]

All versioning and tracking elements are related to the challenge of *Reproducibility*, which cannot be fully achieved without them. In addition, *Model Verification* is facilitated by the enhanced comparison capabilities achieved through experiment tracking. The improved verification leads to fewer flaws and thus better predictive performance, which increases the *Trust of End-Users*.

5.1.5 Establish a Sophisticated Model Serving Strategy

Machine learning systems are usually regular applications that are extended by features provided by machine learning models. This leads to several ways the machine learning model can be integrated into the application. There is no universally valid recommendation for all use cases, meaning that the appropriate strategy must be determined case-by-case. The decision depends on factors such as the level of interaction between the application and model development teams, the type of devices the application is running on, and the characteristics of the environment in which the machine learning model will be operated. Nevertheless, a clear strategy definition for deploying machine learning models is essential for a holistic MLOps architecture. Salama et al. (2021) recommend that, in principle, both low-latency, near-real-time (online) predictions and high throughput batch (offline) predictions should be supported. The authors also suggest that common machine learning frameworks should be supported, as well as composite predictions where multiple models are executed hierarchically or simultaneously and aggregated into a single output.

[231] Cf. Salama et al. (2021).

[232] See Hazelwood et al. (2018).

Serving Strategies can be divided into four groups:

- *Embedded Model*: The model is viewed as a dependency built and packaged together with the application code.[233] This approach has advantages in terms of latency since no external services need to be retrieved for predictions. However, the flexibility of the application and model development teams is limited, as the tight interdependencies require much coordination. In addition, the scaling requires the entire application to be deployed multiple times, which results in a significant resource overhead. The approach is particularly suitable for applications installed on the end device that need to make local predictions, for instance, to ensure privacy.
- *Model as Data*: This strategy provides the model independently of the application and the application incorporates the model at runtime.[234] This provides the same latency benefits as the embedded model solution, except for the initial load time, and brings the additional flexibility benefit of independent release cycles between model and application development. However, architectural changes that impact the use of the model or changes in the runtime dependencies associated with the model still require close cooperation between the development teams. As a result, scalability is only marginally improved in cases where multiple models are used at different points in time and resources can be saved through intelligent model management.
- *Model as Microservice*: In this approach, the model is deployed independently of the application and requested via remote invocations such as REST[235] or gRPC[236] for each prediction.[237] This deployment type is widespread because it has significant advantages in terms of scalability.[238] In addition, new model versions can be deployed entirely independently of the application development teams if nothing has changed in the interface definition. However, remote invocations inevitably lead to increased latency.[239]

[233] Cf. Sato et al. (2019).

[234] Cf. Sato et al. (2019).

[235] REST is the abbreviation for REpresentational State Transfer and describes a universal API standard that utilizes HTTP requests.

[236] gRPC is a Remote Procedure Call protocol developed by Google that utilizes protocol buffers instead of JSON serialization to achieve performance benefits.

[237] Cf. Sato et al. (2019).

[238] See, e.g., Charrington (2020); Karamitsos et al. (2020); Lakshmanan et al. (2020).

[239] Cf. Sato et al. (2019).

- *Model as Serverless Function*: For this approach, the model is also deployed independently of the application and invoked by remote calls for predictions, but it is provided as a function in a serverless computing environment. Serverless computing environments abstract all infrastructure management from the user and allow automatic scaling.[240] While container orchestration systems such as Kubernetes often offer auto-scaling capabilities as well, this usually requires at least one container to run permanently. The main drawback with serverless architectures is increased latency, which can be critical in many real-time applications. Benchmarks indicate that even without cold start overhead, the overhead compared to traditional microservice architectures can range from ten to forty-five percent of the end-to-end latency of a query.[241] The cold start overhead is the additional latency caused by the initial start of a new container for the execution of a function.[242] Li et al. (2020) suggest that traditional microservice architectures are preferable for applications with high user interaction, while serverless architectures are best suited for applications with low utilization.

Table 3 compares the serving strategies, with the scale ranging from extremely poor (- -) to exceptionally good (+ +) as an overview. Scalability refers to the overhead that arises from scaling in the sense of deploying multiple units. Low overhead leads accordingly to good scalability. Independence refers to the degree of communication and integration required between different teams with their deployments. Increased independence thus facilitates agile ways of working and allows shorter release cycles. Latency indicates the relative end-to-end latency for prediction queries. The latency rating is also a proxy for the independence of network requests. Thus, a good latency rating indicates a lower vulnerability to related network errors.

Table 3: Serving Strategy Comparison

Strategy	Scalability	Independence	Latency
Embedded Model	- -	- -	+ +
Model as Data	-	-	+
Model as Microservice	+	+ +	-
Serverless Model	+ +	+ +	- -

[240] Cf. Lloyd et al. (2018).

[241] Cf. Li et al. (2020).

[242] Cf. Bac et al. (2022).

A mature service strategy mitigates the impact of *Heterogeneous Skill Levels* by streamlining processes and reducing internal dependencies. The challenges arising from the *Lack of Standards* can also be reduced by clearly defined deployment workflows for various machine learning frameworks.

5.1.6 Facilitate Online Experimentation for Model Verification

Verifying the predictive performance of machine learning models in production environments before the deployment is extraordinarily complex due to their black-box characteristics. Even large evaluation datasets cannot guarantee that they cover all live scenarios or that the environment has not already changed. It is therefore essential to provide convenient options to assess new models with live data before replacing the old model.

Online Experimentation refers to the evaluation of models using data from real-world operations. There are two categories of online experiments:

- *Shadow Deployments*: The traffic from the production environment is also sent to the new model, while all users continue to receive predictions from the old model.[243] This allows a direct performance comparison between the models before the new model is promoted. The risk of bothering users with bugs caused by the new model is minimized accordingly, but shadow deployments also lead to duplicated computational effort and delayed deployments.
- *Competing Models*: The new model is deployed in addition to the old model, and a subset of the traffic is routed to the new model. This category of strategies includes, for instance, A/B testing and canary deployments. However, these are more complex to implement than shadow deployments due to the additional routing overhead. In addition, it may take some time to obtain sufficient data for a statistically significant evaluation.[244] Another competing model strategy Sato et al. (2019) mentioned is the multi-armed bandit, where multiple models are deployed in parallel, and an increasing share of traffic is diverted to the best-performing model using an appropriate reward function.

Reviewing models with live production data allows the identification of *Training-Serving Skews* before they have a major negative impact. Additionally, online experiments extend the

[243] Cf. Sato et al. (2019).

[244] Cf. Sato et al. (2019).

opportunities for *Model Verification* since indirect effects in the overall system can be measured besides the actual predictive performance. Furthermore, the *Trust of End-Users* is continuously strengthened by ensuring that the latest models consistently perform better than the old ones.

5.1.7 Facilitate Data Pre-Processing Through Processes and Tools

The success of machine learning systems depends on the quality and quantity of the available data. Although a narrow definition of MLOps might suggest that acquiring and preparing sufficient high-quality data is more commonly included under the term DataOps, this would not reflect the reality of many organizations. As highlighted in a previous chapter, the biggest marginal gain is almost always in the data; making clean data accessible is much more important than minor model improvements.[245] In other words, the optimizations and automation achieved through MLOps will not lead to the desired results if data quality and quantity are insufficient. It is therefore essential to take a holistic view of machine learning system development and, if not already in place, to establish structures for efficient data preparation and analysis.

Labeling Processes directly impact prediction quality and are very time-consuming when performed manually. A strictly controlled process for the generation of labels is therefore essential to ensure label accuracy.[246] In manual labeling, annotation tools should be utilized to speed up the processes and reduce the number of errors.[247] There are various more sophisticated approaches based on semi-supervised learning or weak labels, but they often do not match the quality of data labeled by domain experts.[248] The semi-supervised learning approaches attempt to infer the labels of the remaining data from labels created for a subset of the dataset. In the weak label approach, sufficient accuracy is sought by matching many inaccurate labels that may be obtained more efficiently. Roh et al. (2021) point out that external resources can also be leveraged to obtain the data labels, e.g., through crowdsourcing.

Data Visualization is crucial in identifying areas in which quality issues exist or where more data is needed. Therefore, there should be an interactive visualization capability in the form of dashboards that allow exploratory data analysis.[249] In addition, an overview of potential

[245] Cf. Tagliabue (2021).

[246] Cf. Serban et al. (2020).

[247] Cf. Ruf et al. (2021).

[248] Cf. Roh et al. (2021).

[249] Cf. Tagliabue (2021).

problems or errors in the data set and visualizations such as distributions of attributes or possible alterations of individual variables over time should be given.[250] Besides the aspects related to the general predictive performance, special attention should be paid to fairness-related features such as race, gender, or age.[251]

Efficient data labeling processes are beneficial in diminishing the challenges of *Data Labeling* and the *Lack of Data*. Additionally, the structured process can increase *Data Quality* and facilitate *Regulatory Compliance*. Data visualization also assists in the early detection of quality deficiencies and their appropriate correction. In addition, imbalances can be identified more quickly and thus *Model Fairness* criteria verified. Furthermore, the visual comparison of training and production data also helps to decrease the likelihood of *Training-Serving Skews*.

5.2 Open-Source Tools

The introduction of MLOps confronts organizations with new challenges that require corresponding software solutions. This demand has led to many OSS projects that allow companies to build sophisticated MLOps platforms without depending on third-party vendors. However, no existing OSS covers all MLOps capabilities as described in chapter 2.4.3.[252] Accordingly, a holistic architecture is always composed of various tools which must be chosen according to individual requirements. Hybrid solutions may also be a viable option where specific products from public cloud providers complement self-hosted OSS.

To facilitate the development of an MLOps platform suitable for the individual use case, a variety of OSS solutions are highlighted in the following. All the information about the license, the stars, and the launch date were retrieved from the respective GitHub repositories on the 14th of June 2022. As described in chapter 2.4.1, MLOps builds on the foundation of existing DevOps capabilities, explaining why these are prerequisites for many of the tools. A complete listing of the tools highlighted in this thesis and links to the GitHub repositories can be found in Table 22 in the appendix.

Only tools that have received at least one thousand stars on GitHub and a version update in the last twelve months were considered. GitHub stars are a sign of appreciation and can be awarded to repositories by logged-in users. They thus serve as a proxy for the size of the community

[250] Cf. Ruf et al. (2021).

[251] Cf. Sato et al. (2019).

[252] Cf. Ruf et al. (2021)

supporting the corresponding open-source project. Since the community of open-source projects is actively involved in the development, the likelihood of quick bug fixes increases with the community size. Therefore, the restriction to projects with at least one thousand stars aims to ensure a minimum level of software quality. Furthermore, the restriction to projects with an update in the last twelve months is intended to ensure that the product is still under active development.

Some characteristics were described in more detail for up to three tools in each category. For this purpose, the tools with the most GitHub stars were selected to highlight the most relevant tools from a community perspective. The limitation to three instruments is a trade-off between the scientific added value and the practical utility of this work. This should facilitate the identification of relevant tools for further analysis of their suitability in the specific use case.

5.2.1 Data Visualization

Grafana: Platform for visualization of data in the form of live dashboards. It is often used in infrastructure monitoring but is very flexible due to the wide range of supported databases and extensibility through plug-ins. In addition, it supports alerting to a variety of notification or messaging systems.

Apache Superset: Data visualization platform designed for large datasets. Features a wide range of compatible database systems, a customization API, and a web-based graphical user interface that can be extended with plug-ins to create customized visualizations. In addition, it supports alerting when threshold values are exceeded.

Facets: A data visualization tool for tabular data specifically designed for machine learning systems. Implemented as Python library for the execution from computational notebooks. Features two visualization options, a general overview for detecting unexpected feature values, missing features, and training-serving skews as well as an interactive view for exploratory analysis.

Table 4: Open-Source Data Visualization Tools

Name	Launch	Stars	License	Remark
Grafana	2013-12-11	49467	AGPL-3.0	Web-based platform
Apache Superset	2015-07-21	46590	Apache-2.0	Web-based platform
Facets	2017-07-07	6866	Apache-2.0	Python library
Evidently	2020-11-25	2416	Apache-2.0	Python library

5.2.2 Data Labeling

Label Studio: A very versatile tool for a wide range of data types with substantial customizability and offers one-click deployments for Heroku[253], Azure[254], and Google Cloud[255], as well as a pre-built docker-compose script.

doccano: Annotation tool designed explicitly for Natural Language Processing (NLP). Especially suitable for Named Entity Recognition, Sentiment Analysis, Translation, Text to SQL, and Speech to Text applications, but also supports simple image classification.

Rubrix: Another tool specially designed for NLP applications with excellent documentation and built-in support for sophisticated features such as active learning. Features integrations for automated model monitoring of major NLP libraries such as spaCy[256], Hugging Face[257], and FlairNLP[258].

Table 5: Open-Source Data Annotation Tools

Name	Launch	Stars	License	Remark
Label Studio	2019-06-19	9234	Apache-2.0	Supports images, audio, text, and time series
doccano	2018-05-09	6328	MIT	Supports only text
Rubrix	2021-04-28	1128	Apache-2.0	Supports only text

[253] https://www.heroku.com

[254] https://azure.microsoft.com

[255] https://cloud.google.com

[256] https://spacy.io

[257] https://huggingface.co

[258] https://github.com/flairNLP/flair

5.2.3 Data Versioning

Dolt: Database system with similar functionalities as a SQL database but extended with version control that follows a similar logic as git. Access rights are managed as in MySQL, there is a backup creation command, and replication capabilities are built-in.

Git LFS: Allows to replace large files from the git code repository with pointers and stores them on servers dedicated to large files. The advantage of this is that all data types can be versioned with the usual git workflow. However, managing the folder structure can become challenging with substantial amounts of data.

DVC: Command Line Interface (CLI) with similar logic as Git LFS. It has the same advantages and disadvantages but offers more freedom in choosing the location for large files by utilizing any S3-compatible storage[259].

Table 6: Open-Source Data Versioning Tools

Name	Launch	Stars	License	Remark
Dolt	2019-07-24	10757	Apache-2.0	SQL database with version control
Git LFS	2013-09-22	10375	MIT	Extends git repositories
DVC	2017-03-04	9842	Apache-2.0	Extends git repositories
Pachyderm	2014-09-04	5511	Custom	Provides data versioning and pipelines
ClearML	2019-06-10	3230	Apache-2.0	Multipurpose machine learning framework
lakeFS	2019-09-12	2522	Apache-2.0	Provides a git like interface
TerminusDB	2019-07-23	1835	Apache-2.0	Document-oriented graph database
Quilt	2017-02-10	1156	Apache-2.0	Versioning system for S3 buckets

5.2.4 Feature Stores

Pachyderm: Combines data pipelines with data versioning and offers excellent scaling capabilities in Kubernetes. Both structured and unstructured data are supported, and integration for computational notebooks is built-in.

FEAST: Provides transformations and the option of separate offline and online stores. While offline stores are designed for substantial amounts of data, online stores allow near-real-time

[259] Amazon S3 is an object storage system whose interface has been adapted by many third-party solutions.

access. FEAST expects for each model to have its own feature service, which should be immutable and therefore offers no versioning.

Table 7: Open-Source Feature Stores

Name	Launch	Stars	License	Remark
Pachyderm	2014-09-04	5511	Custom	Provides data versioning and pipelines
FEAST	2018-12-10	3265	Apache-2.0	Provides only transformations

5.2.5 Pipelines

Apache Airflow: Workflows for Apache Airflow are defined in Python and can be managed via a web interface. It is designed for use cases with mostly static workflows that evolve slowly. A mode for local debugging is build-in, and various integrations for third-party tools exist.

Kubeflow: Specifically designed for machine learning development and therefore extends workflows with features such as experiment tracking and artifact storage. Workflows are also defined in Python and can be managed via a web interface. However, the available integrations are more specific to machine learning use cases, making Kubeflow less versatile than Apache Airflow.

Argo Workflows: Workflows are designed in YAML and are intended to be executed in Kubernetes. Like Apache Airflow, Argo workflows is a general-purpose workflow tool that can orchestrate tasks such as infrastructure automation and CI/CD workflows in addition to data and machine learning pipelines. Argo focuses on parallelization and allows more freedom for the orchestration in Kubernetes.

Table 8: Open-Source Pipeline Tools

Name	Launch	Stars	License	Remark
Apache Airflow	2015-04-13	26334	Apache-2.0	Deployment in Kubernetes cluster or executes in Celery
Kubeflow	2017-11-30	11578	Apache-2.0	Deployment in Kubernetes cluster
Argo Workflows	2017-08-21	11197	Apache-2.0	Deployment in Kubernetes cluster
Prefect	2018-06-29	9272	Apache-2.0	Deployment in Kubernetes cluster or to a single node
Kedro	2019-04-18	7291	Apache-2.0	Python library
MetaFlow	2019-09-17	5700	Apache-2.0	Deployment in Amazon Web Services (AWS)
Pachyderm	2014-09-04	5511	Custom	Deployment in Kubernetes cluster
dagster	2018-04-30	4871	Apache-2.0	Deployment in Kubernetes cluster or any container-based orchestration system
CML	2020-02-26	3333	Apache-2.0	Command line interface
ClearML	2019-06-10	3230	Apache-2.0	Deployment in Kubernetes cluster or to a single node
Polyaxon	2016-12-26	3104	Apache-2.0	Deployment in Kubernetes cluster
Ploomber	2020-01-20	2435	Apache-2.0	Python library
Flyte	2019-10-21	2399	Apache-2.0	Deployment in Kubernetes cluster
ZenML	2020-11-19	2098	Apache-2.0	Python library
TensorFlow TFX	2019-02-04	1771	Apache-2.0	Python library

5.2.6 Model Registry and Experiment Tracking

MLflow: Platform for experiment tracking and model management via API or web interface. Offers a wide range of integrations and thus great flexibility in the design of MLOps architectures.

Scared: Python library to track experiments locally, in a database, S3-compatible storage, or messenger channel. In contrast to MLflow, there is no hosted service, and visualizations must be realized using third-party tools.

ClearML: Platform that additionally provides many other MLOps functions such as data versioning, pipelines, or model serving. One of the platforms that comes close to an all-in-one solution, but at the cost of customizability.

Table 9: Open-Source Model Registry and Experiment Tracking Tools

Name	Launch	Stars	License	Remark
MLflow	2018-06-05	12033	Apache-2.0	Experiment tracking and model registry
Sacred	2014-03-31	3838	MIT	Only experiment tracking
ClearML	2019-06-10	3230	Apache-2.0	Multipurpose machine learning framework
Polyaxon	2016-12-26	3104	Apache-2.0	Multipurpose machine learning framework
Aim	2019-05-31	2523	Apache-2.0	Only experiment tracking

5.2.7 Model Serving

Ray: Framework for distributed computing in Python, which provides a library for model serving as well as libraries for data processing, distributed training, hyperparameter optimization, and reinforcement learning. Particularly suitable for use cases with extensive scaling requirements.

Cortex: A serving tool with integrated load balancing, autoscaling, and monitoring interface, but designed only for deployments in AWS.

Triton Inference Server: Serving framework with various optimizations for GPU inference, but also supports CPU inference. Models are loaded from a model registry on demand and are not permanently integrated into the container. Provides an interface for monitoring but no integrated load balancing or autoscaling features.

Table 10: Open-Source Model Serving Tools

Name	Launch	Stars	License	Remark
Ray	2016-10-25	20913	Apache-2.0	Multipurpose machine learning framework
Cortex	2019-01-24	7756	Apache-2.0	Built for AWS
Triton Inference Server	2018-10-04	3737	BSD-3-Clause	Multi model serving framework
BentoML	2019-04-02	3541	Apache-2.0	Model containerization framework
ClearML	2019-06-10	3230	Apache-2.0	Multipurpose machine learning framework
Seldon Core	2017-12-20	3205	Apache-2.0	Includes online experimentation capabilities
TorchServe	2019-10-03	2653	Apache-2.0	Built for PyTorch models
Opyrator	2021-04-06	2637	MIT	Creates REST APIs for Python functions
TensorFlow TFX	2019-02-04	1771	Apache-2.0	Built for TensorFlow models
KServe	2019-03-27	1538	Apache-2.0	Built for Kubernetes

5.2.8 Other Related Tools

Prometheus: Metric store for time series data that works based on a pull concept, where Prometheus regularly queries metrics of services. Provides a push gateway for short-lived services and leverages Grafana for dashboards and alerts.

Apache Spark: A very versatile execution engine for scaling any data processing operations. Provides integrations for various machine learning frameworks and supports Python, SQL, Scala, Java, and R.

Horovod: A distributed training framework that supports TensorFlow, Keras, PyTorch, and Apache MXNet.

Table 11: Other Related Open-Source Tools

Name	Launch	Stars	License	Remark
Prometheus	2012-11-24	42935	Apache-2.0	Metric store for time-series data
Apache Spark	2014-02-25	33130	Apache-2.0	Distributed data processing tool
Horovod	2017-08-09	12491	Apache-2.0	Distributed DL training tool
Dask	2015-01-04	9985	BSD-3-Clause	Python library for distributed computing
JupyterHub	2015-04-09	9124	BSD-3-Clause	Spawner for computational notebooks
MindsDB	2018-08-02	7639	GPL-3.0	In-Database machine learning tool
DeepSpeed	2020-01-23	6884	MIT	Distributed DL training / inference tool
Great Expectations	2017-09-11	6715	Apache-2.0	Tool for data validation, documentation, and profiling
Optuna	2018-02-21	6525	MIT	Hyperparameter optimization tool
Hyperopt	2011-09-06	6255	Custom	Distributed hyperparameter optimization tool
TensorBoard	2017-05-15	5879	Apache-2.0	Visualization toolkit for TensorFlow
H2O-3	2014-03-03	5850	Apache-2.0	Partly open-source analytics platform
dbt	2016-03-10	4998	Apache-2.0	Data transformation tool
Nuclio	2017-05-19	4419	Apache-2.0	Serverless framework for data science
River	2019-01-24	3435	BSD-3-Clause	Online machine learning framework
Accelerate	2020-10-30	2547	Apache-2.0	Facilitates multiple GPU support in PyTorch
Ivy	2021-01-19	2442	Apache-2.0	Interoperability framework
Scikit-Optimize	2016-03-20	2350	BSD-3-Clause	Hyperparameter optimization tool
Talos	2018-05-04	1525	MIT	Hyperparameter Optimization for TensorFlow, Keras and PyTorch
Petastorm	2018-06-15	1441	Apache-2.0	Distributed DL training tool
whylogs	2020-08-14	1372	Apache-2.0	Logging library for any kind of data
Katib	2018-04-03	1182	Apache-2.0	Hyperparameter optimization tool for Kubernetes cluster

6 MLOps Architecture Design

Due to the heterogeneous use cases of machine learning systems and organizations' capabilities, an optimal MLOps architecture for all possible scenarios is not feasible. Therefore, it is essential to prioritize the MLOps technical capabilities based on the business objectives and existing capabilities, as described in chapter 2.4.3. Based on this prioritization, a selection of components can be made to implement the MLOps capabilities incrementally. Potential criteria that may be involved in the selection process include the use case, financial aspects, the existing IT infrastructure, and aspects such as the preference between on-premise and Software as a Service (SaaS) solutions. These values, desires, fears, goals, norms, and budgets are termed problem contexts in design science research and are beyond the researcher's control.[260]

In the subsequent, this thesis will first elaborate on the problem context of the company for which the MLOps architecture was designed. The following subchapter introduces the MLOps architecture design and gives insights into the decision-making process. This is followed by a subchapter in which the findings of the conducted focus group to validate the design are outlined. Finally, the last subchapter highlights the changes to improve the design that resulted from the validation.

6.1 Problem Context

The company for which the design was developed is a medium-sized software manufacturer offering solutions based on machine learning to enterprises in service controlling, planning, operations, and design. The application areas of machine learning models are diverse and range from financial planning based on time series data to NLP tasks for support services. All solutions based on machine learning are provided in the form of SaaS and are operated by the company itself. However, different teams are responsible for the different services, and the services are operated for various companies.

The company operates its own data centers for the IT infrastructure and supplements them as needed with resources provided by Infrastructure as a Service (IaaS) providers. There is a strong focus on on-premise solutions for internally used software solutions, so third-party SaaS products are only used in exceptional cases. The foundation for the operation of products that are used internally and the products that are hosted for customers is thereby provided by the

[260] Cf. Wieringa (2014).

HashiCorp tech stack.[261] This means that Terraform is used to control the infrastructure, networking is handled by Consul, authentication is controlled by Vault, and the applications are orchestrated in a Nomad cluster.

In the management of the development teams within the company, strong emphasis is placed on the principles of agile working. In addition to short release cycles, strong customer orientation, and other aspects, this includes that teams work in a self-organized manner.[262] The machine learning development teams build the products almost entirely from scratch, up to the definition of the API endpoints. However, the teams use sophisticated DevOps structures that exist within the company. Regarding the MLOps-specific tools, only one of the machine learning teams utilizes MLflow to manage their experiments. There is no online experimentation capability yet, and the monitoring of models in production and the corresponding data does not follow a standardized process. Most data processing and experiments are performed locally or via a shared Jupyter notebook[263] server. Azure Pipelines[264] are used for DevOps-related tasks, while a solution for data and machine learning pipelines is not established. Azure Repos[265] are used for code versioning, while no solution exists for data versioning. MinIO[266] is utilized for S3-compatible object storage, and PostgreSQL[267] serves as database system. Grafana is used for the visualization of performance metrics and alerting.

6.2 Composition of the Architecture

The core principles of MLOps, like those of DevOps, are automation and reusability. Accordingly, the missing pipeline solution was the first focus during the design of the MLOps architecture. The challenge here was that most of the OSS tools available on the market are designed for Kubernetes clusters, while Nomad clusters are not supported. So that despite intensive research, no solution could be found that natively supports scaling in Nomad clusters or provides documentation for it. Finally, dagster was chosen as it provides extensive freedom for implementing customized deployments.

[261] https://www.hashicorp.com/

[262] See Beck et al. (2001).

[263] The Jupyter project (https://jupyter.org) is a non-profit open-source project that provides among other tools solutions for computational notebooks.

[264] https://azure.microsoft.com/en-us/services/devops/pipelines/

[265] https://azure.microsoft.com/en-us/services/devops/repos/

[266] https://min.io/

[267] https://www.postgresql.org/

dagster describes itself as a data orchestration platform and defines workflows as DAGs. The individual operations of the DAG are defined in Python and assigned with decorators so that developers do not need to change their workflows significantly. These operations are combined with additional functions to form graphs in which the operation functions are nested. This allows the simple reusability of operations in multiple graphs and the nesting of multiple subgraphs. Graphs can then be parametrized to create jobs based on them. This allows adapting the parameters of the graph to different environments, e.g., production environments and the local system, for debugging and testing purposes. In addition to defining DAGs, schedules for execution and sensors for triggering specific pipelines are also defined in Python. Therefore, all relevant settings and workflows can be versioned with existing code versioning systems without additional overhead.

DVC was chosen for data versioning because its git-style versioning makes it usable for all data types, and the ability to choose the storage backend fits into the existing on-premise architecture of the company. Additionally, DVC does not require any restructuring of the existing database systems or their access, such as the hosted solutions might have required. Furthermore, the similarities to git allow developers to get used to DVC quickly.

MLflow was chosen for experiment tracking and model management because it offers many integrations and thus does not cause a lot of lock-ins concerning other MLOps tools. Furthermore, it offers options to group experiments and natively supports all major machine learning frameworks. The experiments can be conveniently compared via a graphical user interface, which also allows the management of models. APIs exist for Python, R, and JAVA, as well as a REST API. In addition, there is an integration for dagster, which simplifies experiment tracking based on pipeline runs. Finally, the fact that one team is already using some features of MLflow further reduces the adoption barriers and allows for a quick expansion to other teams.

Triton Inference Server was chosen to serve the models because it supports the deployment of models of all major machine learning frameworks and provides an extension for the interaction with MLflow. Furthermore, it supports GPU and CPU inference as well as features like dynamic batching and model ensembles. In addition, an interface to export metrics for Prometheus is provided for monitoring. Unlike other serving frameworks such as BentoML, the models are not permanently integrated into the container but loaded from a model repository as needed. Due to many natively supported machine learning frameworks and its extensibility via customized backends, Triton Inference Server should fit the requirements of all teams.

Prometheus was the choice for storing metrics due to the native support of the Triton Inference Server. In addition, the company uses various tools whose monitoring can be facilitated by Prometheus, such as Nomad, MinIO, RabbitMQ, and PostgreSQL.

Grafana was chosen to visualize the metrics collected, which was another straightforward decision due to its versatility. Besides the great extensibility and adaptability, a decisive reason was that Grafana is already used in various company applications so that other solutions would have unnecessarily increased the complexity of an existing IT landscape.

The chosen architecture builds on existing DevOps systems by using Azure Repos for code versioning, MinIO as object storage for DVC, MLflow, and the Triton Inference Server Model Repository, as well as PostgreSQL databases for dagster, MLflow, and Grafana, in addition to the Nomad cluster for the deployed services.

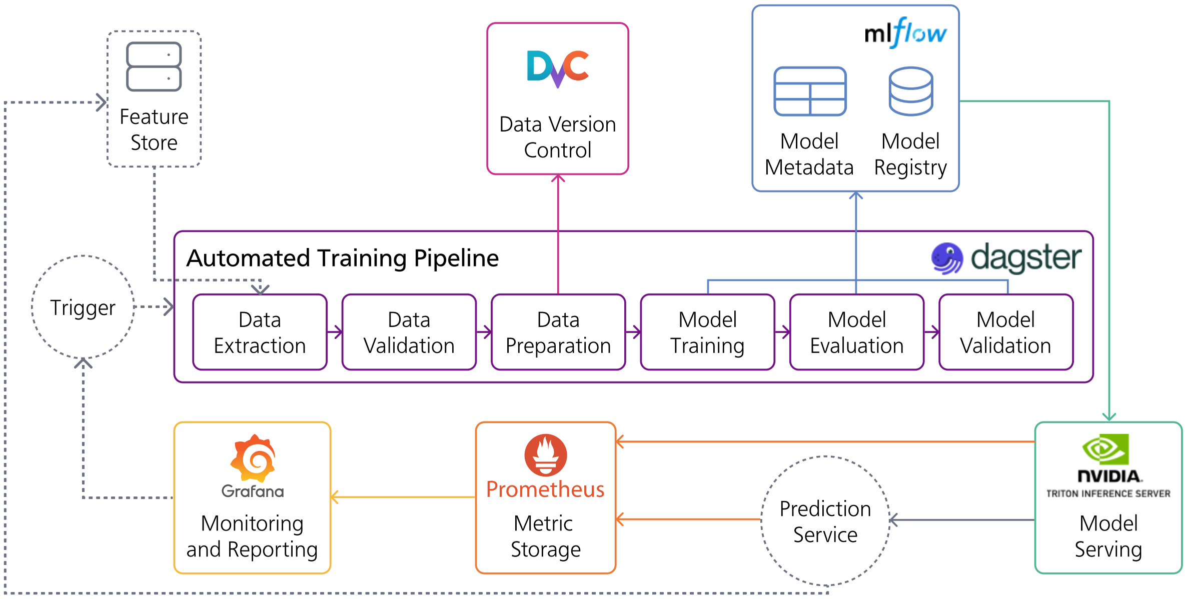


Figure 13: MLOps Architecture

The relationships between the components within the resulting MLOps architecture are shown in Figure 13. It represents an exemplary process for a supervised learning use case with continuous training. In particular, concerning the pipeline presented for dagster, several different pipelines will be required for real-world applications. The gray-dashed components are not explicitly specified by the architecture and are determined dynamically by the use case of the teams.

Prediction Service in this context refers to the application which retrieves the model predictions using remote invocations and provides them to the end-user. This depends entirely on the use case and, for example, could be a chatbot that utilizes an NLP model to generate the answers.

Triggers describe the conditions that must be met to trigger retraining of the model as part of the continuous training. They are based on the data and the model monitoring in the production system, for example when drifts are detected.

The architecture design focused on the core elements of MLOps so that DataOps processes are not represented. The *Feature Store* was viewed as the final element of the DataOps processes and thus not explicitly defined in the design. This only reflects the focus of the third RQ of this thesis and does not constitute an assessment of importance. As emphasized in the previous sections, MLOps processes can only succeed if sophisticated DataOps and DevOps structures are in place.

6.3 Focus Group Validation

The validation of design artifacts always arises from the interaction of the design artifact within the specific problem context in which the validation is performed.[268] It is therefore vital to understand that the following validation results are not necessarily applicable to other problem contexts.

The focus group revealed that the participants expect all architecture components to provide significant added value. The most tremendous potential was seen in *dagster* and *MLflow*, although two data scientists highlighted model serving. The ability to execute independent pipeline steps in parallel, the improved understanding of workflows of other teams, and the definition of operations as Python functions were highlighted as particularly positive concerning *dagster*. The outstanding characteristic of *MLflow* was the tracking of experiments to identify the impact of modifications. The group also confirmed that *Prometheus* supported all the relevant metrics and saw *DVC* as a valuable complement to the rest of the architecture.

Grafana, as a tool for visualization and alerting, was a component that did not require much discussion due to the existing intensive use in the company. Several participants commented, however, that they view the MLOps architecture as a behind-the-scenes system and do not expect reporting to external parties from it.

[268] See Wieringa (2014).

The biggest issue related to the architecture, according to the participants, is the lack of authentication capabilities. Grafana is the only component of the architecture that natively supports user logins. According to the group, it is challenging to establish role-based authentication if it is not natively supported by the tool. One participant explained that there might be non-disclosure agreements restricted to the team working on the project in individual projects. To address this challenge, the group proposed the total isolation of environments for customers with exceptionally high privacy requirements.

The *Triton Inference Server* was also assessed critically due to its inherent design where models are dynamically loaded and unloaded from a model store, significantly increasing the scaling and load balancing complexity. Furthermore, it was emphasized that GPUs currently do not play any role in the model serving of the company and will not do so in the near future.

The MLOps architecture was overall widely supported by all participants. However, it was pointed out that additional tools always bring additional complexity and that it is therefore always necessary to critically examine which components are necessary.

6.4 Architecture Revision After Validation

The defined MLOps architecture was revisited critically based on the focus group discussions. The most significant modification concerns the model serving solution where BentoML has replaced the Triton Inference Server. The decision was explicitly driven by the concerns about the increased complexity of the scaling and load balancing of the Triton Inference Server. In contrast to the concept of the Triton Inference Server, BentoML generates a separate container image for each model. This has the advantage that scaling can be easily controlled by the number of running containers of the specific model, as typical for microservices. BentoML may not solve the issue of not having a native authentication capability, but it does simplify the isolation of models with increased security or privacy requirements. Figure 14 illustrates the architecture after the corresponding adjustment.

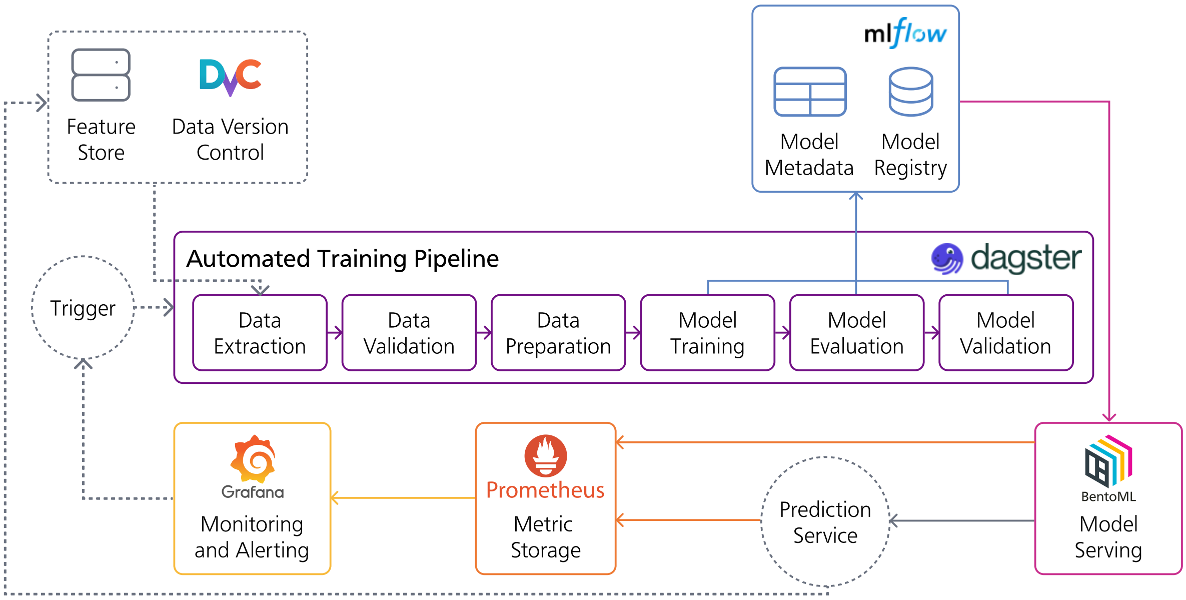


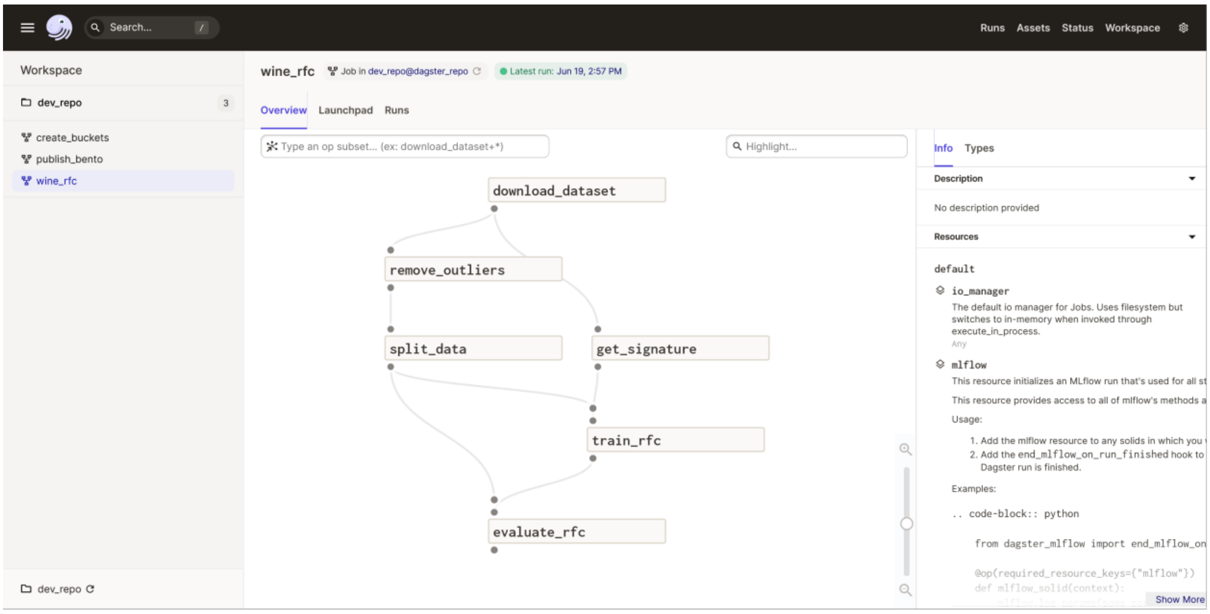
Figure 14: MLOps Architecture After Revision

Apart from the changes described above, only the position of DVC has been adjusted, and the title of Grafana was adapted. The former was changed since the dataset used for pipeline execution is only created by the training pipeline in exceptional cases and has usually been defined in advance. Accordingly, the usage of DVC is instead for identifying the dataset to be extracted and not generating a new data version within the training pipeline. The last adjustment reflects the comments that MLOps is seen as a behind-the-scenes system rather than for reporting purposes.

7 Implementation of the MLOps Design Artifact

All deployable components were implemented as part of a local Docker[269] environment to validate the architecture's internal interaction and provide an exemplary illustration.

Figure 15 shows a simple model training DAG in the dagster web interface. Within the DAG, first, the dataset is downloaded and passed to two operations that can run in parallel. In the left-hand path of the DAG, a simple preprocessing takes place first before the data is split into a training and test split. In parallel, the right-hand path derives the signature of the data, namely the schema of the inputs and outputs. The model is then trained and evaluated to be tracked in MLflow along with the signature.



No description provided

default

io_manager

The default io manager for Jobs. Uses filesystem but switches to in-memory when invoked through execute_in_process.

Any

mlflow

This resource initializes an MLflow run that's used for all st

This resource provides access to all of mlflow's methods a

Usage:

1. Add the mlflow resource to any solids in which you
2. Add the end_mlflow_on_run_finished hook to Dagster run is finished.

Examples:

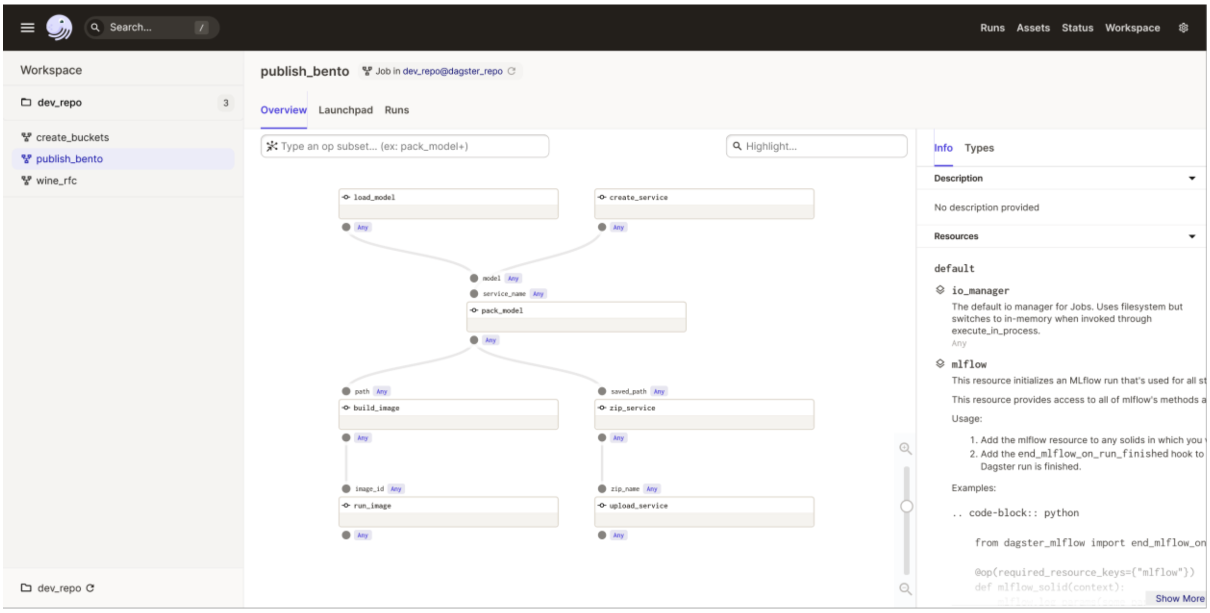
```
.. code-block:: python

    from dagster_mlflow import end_mlflow_on

    @op(required_resource_keys={"mlflow"})
    def mlflow_solid(context):
```

Figure 15: Exemplary Model Training DAG in dagster

Figure 16 presents another DAG that illustrates a deployment pipeline for a model registered in MLflow. The procedure starts by loading the model from MLflow, while the code to create a BentoML service is generated in parallel. The code and model are then packed to build a container image and launch it immediately. In addition, the service is packed into a zip archive and uploaded to S3-compatible storage.

[269] https://www.docker.com

While this illustrates the versatility of dagster, exposing the Docker Daemon to the dagster repository container is required. The exposition of the Docker Daemon allows any Docker container to be launched from the pipelines, posing an unacceptable security vulnerability for many production environments. This emphasizes that the strengths of dagster lie in data and training pipelines, while deployments should still leverage established DevOps systems such as Azure Pipelines.



No description provided

default

io_manager

The default io manager for Jobs. Uses filesystem but switches to in-memory when invoked through execute_in_process.

Any

mlflow

This resource initializes an MLflow run that's used for all st

This resource provides access to all of mlflow's methods a

Usage:

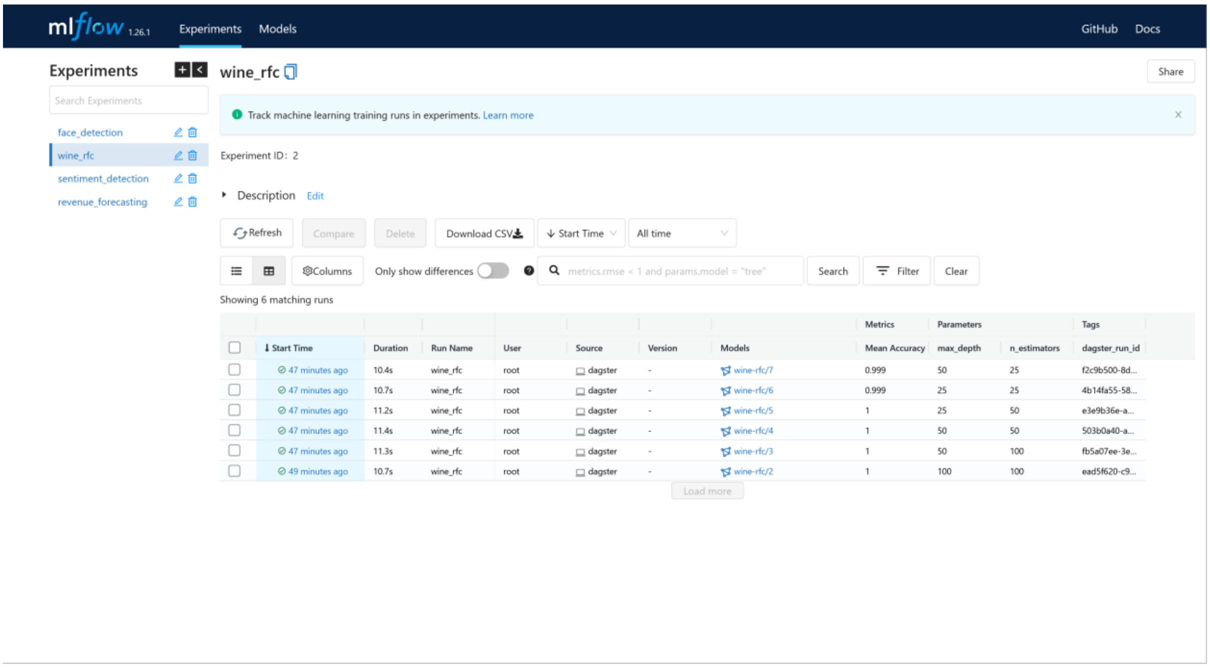
1. Add the mlflow resource to any solids in which you
2. Add the end_mlflow_on_run_finished hook to Dagster run is finished.

Examples:

```
.. code-block:: python

    from dagster_mlflow import end_mlflow_on

    @op(required_resource_keys={"mlflow"})
    def mlflow_solid(context):
```

Figure 16: Exemplary Deployment DAG in dagster

Figure 17 shows the experiment overview of MLflow. In this case, the experiments were all tracked with the MLflow integration of dagster, which automatically passes dagster as the source, the pipeline execution duration, and the unique pipeline run id. Metrics and parameters were passed via the appropriate functions of the MLflow Python library for tracking. The runs with all recorded metrics and parameters can be visualized within MLflow, as shown in Figure 18.



Experiments

Search Experiments

- face_detection
- wine_rfc
- sentiment_detection
- revenue_forecasting

wine_rfc

Track machine learning training runs in experiments. Learn more

Experiment ID: 2

Description Edit

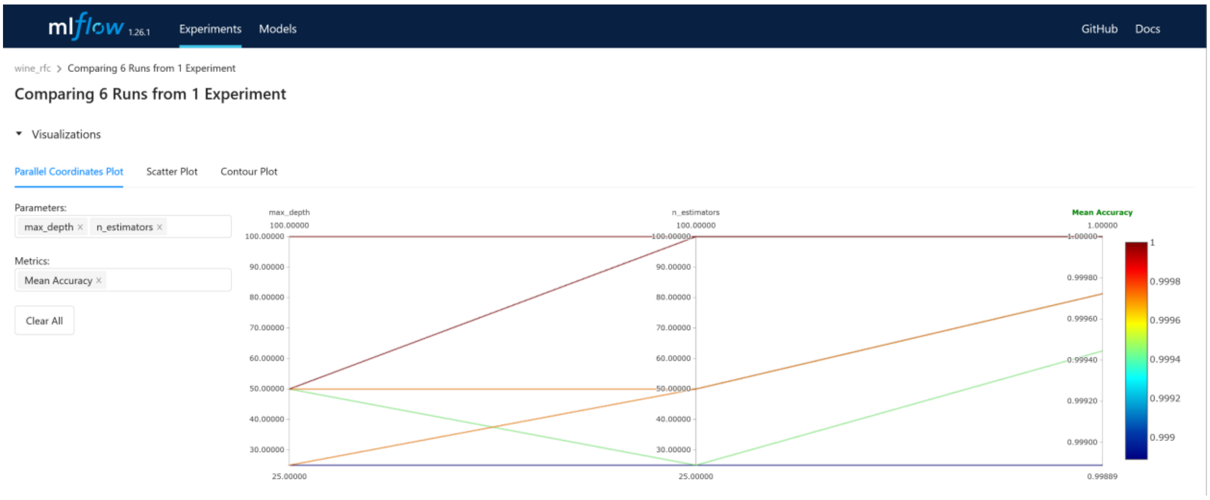
Only show differences

metrics.rmse < 1 and params.model = "tree"

Showing 6 matching runs

								Metrics	Parameters		Tags
□	Start Time	Duration	Run Name	User	Source	Version	Models	Mean Accuracy	max_depth	n_estimators	dagster_run_id
□	47 minutes ago	10.4s	wine_rfc	root	dagster	-	wine-rfc/7	0.999	50	25	f2c9b500-8d...
□	47 minutes ago	10.7s	wine_rfc	root	dagster	-	wine-rfc/6	0.999	25	25	4b14fa55-58...
□	47 minutes ago	11.2s	wine_rfc	root	dagster	-	wine-rfc/5	1	25	50	e3e9b36e-a...
□	47 minutes ago	11.4s	wine_rfc	root	dagster	-	wine-rfc/4	1	50	50	503b0a40-a...
□	47 minutes ago	11.3s	wine_rfc	root	dagster	-	wine-rfc/3	1	50	100	fb5a07ee-3e...
□	49 minutes ago	10.7s	wine_rfc	root	dagster	-	wine-rfc/2	1	100	100	ead5f620-c9...

Figure 17: MLflow Experiment Overview



wine_rfc > Comparing 6 Runs from 1 Experiment

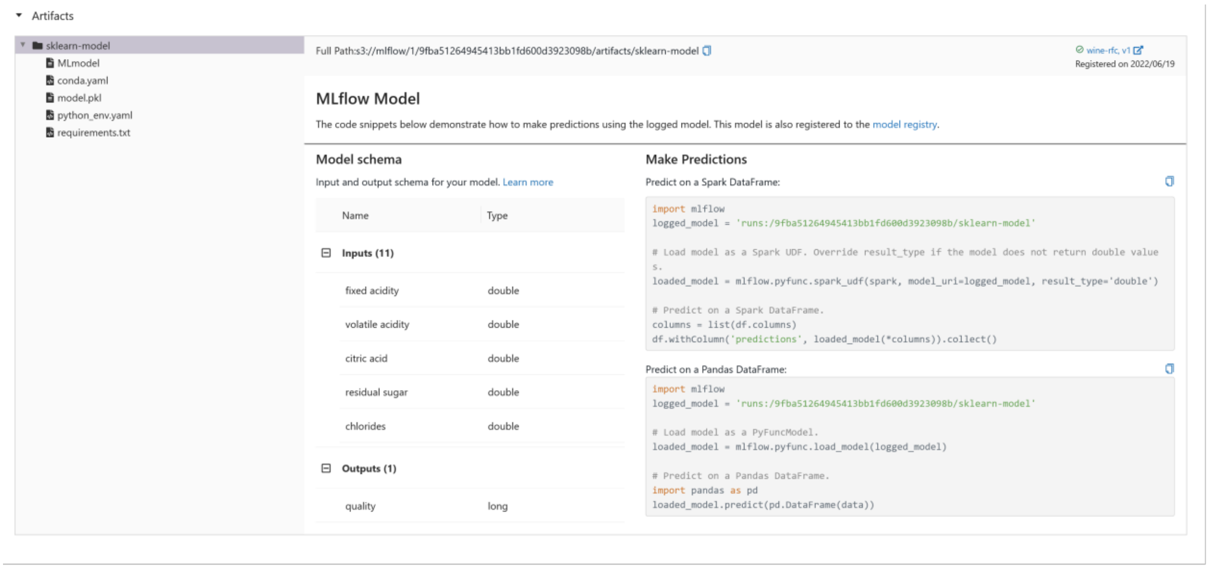
Comparing 6 Runs from 1 Experiment

Parameters: max_depth × n_estimators ×

Metrics: Mean Accuracy ×

Figure 18: MLflow Experiment Run Comparison

In addition, all models resulting from the experiments and tracked in MLflow can be represented in more detail, as shown in Figure 19. In addition to the model artifact and its dependencies, the model signature is displayed here if it was recorded. The detail page thus serves as documentation for each model, which facilitates the usage of the model for practitioners. This is a critical aspect of increasing reusability and exploiting synergies between teams. Furthermore, MLflow provides an overview of models with their deployment status, similar to the one for experiments.



Full Path:s3://mlflow/1/9fba51264945413bb1fd600d3923098b/artifacts/sklearn-model

wine-rfc, v1
Registered on 2022/06/19

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the model registry.

Model schema

Input and output schema for your model. Learn more

Name	Type
⊟ **Inputs (11)**	
fixed acidity	double
volatile acidity	double
citric acid	double
residual sugar	double
chlorides	double
⊟ **Outputs (1)**	
quality	long

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 'runs:/9fba51264945413bb1fd600d3923098b/sklearn-model'

# Load model as a Spark UDF. Override result_type if the model does not return double value
s.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='double')

# Predict on a Spark DataFrame.
columns = list(df.columns)
df.withColumn('predictions', loaded_model(*columns)).collect()
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/9fba51264945413bb1fd600d3923098b/sklearn-model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```

Figure 19: MLflow Model Details

Model containers created with BentoML natively support the publishing of performance metrics for Prometheus. In the local environment, this worked smoothly; however, in production environments, this is slightly more complex as targets for Prometheus need to be dynamically detected when scaling techniques are utilized. However, there are procedures described in the Prometheus documentation for various environments. Data monitoring requires the definition of custom metrics for the corresponding BentoML container, as these depend on the use case. The presentation of metrics stored in Prometheus in Grafana dashboards works seamlessly due to native support by Grafana for Prometheus as a data source.

8 Discussion

The challenges that arise from the productive use of machine learning are multifaceted and, therefore, difficult to assess comprehensively. Furthermore, the extent of the individual challenges and their effects can hardly be generalized and must be examined individually for each case. Nevertheless, the challenges presented in this thesis are likely to apply to many companies, as they were derived from information provided by a substantial number of practitioners. Furthermore, the systematic overview and explanation of common challenges will assist practitioners in developing case-specific solutions. It also increases awareness of challenges that may arise for companies that are just beginning to integrate machine learning into their systems.

Consequently, the best practices presented within this work also vary in relevance depending on the use case. In addition, the best practices highlighted are primarily tailored for supervised learning use cases, and even for this subset of machine learning, completeness cannot be assured. However, they are a valuable contribution when it comes to the design of MLOps solutions, as they reflect the learnings from a large number of practical applications. Furthermore, many best practices apply also to other machine learning methods.

The highlighted open-source tools help gain a broad overview of potential choices for building MLOps capabilities within organizations. However, it is a dynamic environment where new tools regularly emerge, disappear, or new features are added to existing ones. Accordingly, the overview of existing open-source tools loses its informative value over time.

The developed MLOps architecture illustrates the interaction of the individual components based on specific tools selected for one use case. This results in the limitation that the architecture is not necessarily transferable to other tools or use cases. However, the process outlined in the architecture serves as a valuable starting point for adaptation to individual requirements. Furthermore, the tools must be assessed in the context of the specific use case, as all tools involve inevitable trade-offs. The architecture validation also revealed that having all components in place may not always be necessary. Therefore, practitioners must always balance the added value of the tool's features with the added complexity and maintenance requirements it brings. Assessing added value and complexity is also essential to prioritize the MLOps capabilities, as they are usually acquired stepwise.

Existing research on MLOps often relies on empirical methods such as structured and semi-structured interviews. In addition, several user reports and guidelines from individual companies exist. This work builds on this by synthesizing the inherently narrow viewpoints of these sources. Another unique differentiator of this work is the strict focus on OSS solutions for MLOps challenges. Furthermore, the existing scientific literature hardly provides practical guidance on implementing MLOps procedures. Therefore, the results of this thesis represent a bridge between the theoretical principles of the scientific MLOps literature and the practical implementation in organizations.

Moreover, this thesis focuses on the core MLOps activities like model training, model and experiment tracking, model serving, and model monitoring. Other related work, such as that of Munappy et al. (2020), addresses the issue earlier in the process by focusing on DataOps. Furthermore, some publications consider specific problems in detail, such as Tamburri (2020) regarding fairness and explainability in the context of MLOps.

Research validity can be distinguished between construct, internal, conclusion, and external validity.

Construct Validity concerns the extent to which the study design is suitable to address the RQs.[270] This thesis's first two RQs aimed to collect information regarding the practical adoption of machine learning in the organizational context. Due to the broad spectrum of affected companies in combination with the significant medial interest in the topic of machine learning, the synthesis of existing publications is a suitable way to gather the relevant information. The decision to include gray literature in the study is based on the apparent relevance of expert opinions from the industry for the defined RQs. Empirical research methods would be preferable only if information beyond what is already published could be expected, which is unlikely for the above reasons. The third RQ concerns the effectiveness of an MLOps architecture designed as part of this research and validated in a specific context. Validation against existing publications is thus virtually impossible. Among the empirical research methods, the focus group was chosen because MLOps involves a variety of stakeholders, and the interaction between them is critical for determining its effectiveness. While additional insights could have been gained through TAR, this would not have been possible within the time constraints of this work and would have been inefficient for initial validation. The utilized research methods are based on scientifically established procedures described in detail in chapter 3. The threat of insufficient construct validity is correspondingly low for this thesis.

[270] Cf. Kitchenham et al. (2016).

Internal Validity concerns the conduct of data extraction and synthesis and, specifically, whether there are factors that led to a bias in the process.[271] For the conducted MLR, the bias risks are related to selection and instrumentation. Selection biases describe biases induced by the selection of publications included in the MLR.[272] To mitigate this risk, the search terms were broadly defined, multiple search engines were used, clear criteria for inclusion and exclusion of sources were set, and snowballing was applied to extend the pool of sources further. Despite all these measures, the existence of a bias cannot be ruled out with certainty, and completeness can never be guaranteed, especially concerning the gray literature. Instrumentation biases arise from artifacts used to conduct the study.[273] The guidelines for MLRs by Garousi et al. (2019) were strictly followed while creating the data extraction and synthesis spreadsheet to minimize this risk. However, biases cannot be completely ruled out, especially during coding. For the conducted focus group, the selection risk resided in the selection of the participants. This was addressed through a broad selection of participants regarding their role and team affiliation. Similar to the MLR, the risk of instrumentation bias was counteracted by strict adherence to scientifically established procedures. Besides the threats that apply to the MLR, in focus groups, threats arise from the social interaction between the focus group participants.[274] Biases may evolve from the participants' friendships and the different hierarchical levels within the company. The latter poses a non-negligible threat regarding internal validity but could not be avoided due to the small number of eligible focus group participants. Nevertheless, the evaluation of the focus group did not reveal any indications of a resulting bias. Consequently, there is no overall increased risk concerning internal validity.

Conclusion Validity concerns the confidence that the outcomes of an empirical study can be attributed to the treatment.[275] Kitchenham et al. (2016) clarify that in the case of systematic reviews, this refers to the synthesis element and thus differs little from internal validity for any secondary research. Concerning the focus group, there could be low statistical power due to the group size of only eight participants or false expectations caused by uncertainties about the MLOps architecture and its components. However, despite the small number of participants, more than half of the people who would have direct exposure to the proposed MLOps architecture in the company were involved. Furthermore, the threat of false assumptions was

[271] Cf. Kitchenham et al. (2016).

[272] See Wohlin et al. (2012).

[273] Cf. Wohlin et al. (2012).

[274] See Wohlin et al. (2012).

[275] Cf. Kitchenham et al. (2016).

counteracted by an extensive description of the architecture during the focus group and the opportunity to ask questions at any time. Additionally, each conclusion was based on the testimony of at least two participants. Thus, the risk of a lack of conclusion validity is also not significant for this thesis.

External Validity concerns the extent to which the results of a study can be generalized to industrial practice.[276] In secondary research, this refers to assessing the range covered by primary sources regarding their setting, materials, and participants.[277] For the conducted MLR, no exclusions beyond quality control were applied concerning the study design of the primary sources. Correspondingly, the included sources range from individual practitioners' reports to extensive surveys. This was intended to cover a wide range of organizations and therefore allow for generalization. Nevertheless, it cannot be entirely ruled out that specific industries were underrepresented in the pool of sources. For the focus group, external validity can only be assessed with a high degree of certainty for the specific organization with which the validation was conducted. This research cannot determine to what extent this organization represents the AI service provider industry. However, since the proposed MLOps architecture is based on the findings of the broad MLR, a certain degree of generalization is expected. Overall, the risk of insufficient external validity cannot be eliminated entirely, but due to the rigorous scientific methods employed within this thesis, the generalizability of the results is likely.

While no significant validity threats have been identified, there are some natural limitations. Constraints that should be highlighted here are the construct and external validity regarding the third RQ. Thus, implementation within the organization would be desirable for comprehensively validating the MLOps solution, and other AI service providers should be incorporated to ensure generalizability.

[276] See Wohlin et al. (2012).

[277] Cf. Kitchenham et al. (2016).

9 Conclusion

The overarching aim of this study was to facilitate the operationalization of machine learning in production systems. To this end, the problem domain was first analyzed to gain a comprehensive overview of the challenges. Additionally, this thesis synthesized how organizations and practitioners address these challenges and provided a market overview of related open-source tools. Based on this, an MLOps architecture was designed and validated with an AI service provider. Finally, this architecture was implemented as a prototype to allow a more detailed analysis of the components' interaction and provide practical insights. The focus on open-source tools prevents vendor lock-ins and made the results applicable to many practitioners, from individual researchers to large organizations.

Three RQs were covered to fulfill the objective of this thesis. The following is a brief summary of the answers to the questions that emerged from this research.

RQ1: What challenges typically arise with the adoption of machine learning in production systems?

This thesis revealed that the challenges can be divided into three categories: Development, Operation, and People & Compliance. The most common challenges during the *Development* process are the lack of data with sufficient quality and verification of the models against the business objectives. While in O*peration*, the treatment of concept drifts, training-serving skews, and adversarial attacks often cause difficulties. Concept drift refers to supervised learning environments where the relationship between the input data and the target variable shifts over time.[278] Training-serving skews are performance discrepancies due to inconsistencies in terms of data or runtime dependencies between training and serving environments.[279] Finally, challenges related to *People & Compliance* include managing multiple stakeholders with *Heterogeneous Skill Levels*, ensuring *Model Fairness* criteria, and *Regulatory Compliance*. A total of seventeen challenges were identified, resulting from synthesizing sixty-one sources.

RQ2: Which best practices and open-source tools exist to address these challenges?

The most-reported best practices relate to *Model Monitoring*, *Model Training Pipelines*, and *Model Registries*. *Model Monitoring* refers to the continuous monitoring of the model predictions and resource utilization in productive operation, for example, to detect concept drifts or biases. *Model Training Pipelines* allow repeated execution with arbitrary input data by defining fixed

[278] Cf. Gama et al. (2014).

[279] Cf. Salama et al. (2021).

procedures for data preprocessing, feature engineering, model selection, model optimization, and evaluation. They thus form the foundation for the continuous training of models. The *Model Registry* stores all model artifacts with associated metadata such as the model architecture, information about the training dataset, the hyperparameters used, and evaluation results. Fourteen best practices were identified and examined in detail, which emerged from forty-two sources. All these best practices share the common goal of reinforcing the fundamental principles of MLOps regarding automation, reproducibility, and reusability. Sixty-three open-source tools for data visualization, labeling, data versioning, feature stores, pipelines, model registries, model serving, and other related categories have been highlighted. In addition, the most popular three from each category were examined in more detail. An important insight was that none of the tools could manage the entire machine learning lifecycle. Therefore, a holistic open-source solution always requires a composition of several tools. The specific composition of these tools must be assessed case-by-case, as all tools entail inevitable trade-offs.

RQ3: How could an effective MLOps solution be composed of the individual open-source components in the context of an AI service provider?

Based on the problem context of a medium-sized AI service provider, an MLOps architecture was designed. The architecture relied on *dagster* as a pipeline solution, *MLflow* for model registry and experiment tracking, *Triton Inference Server* for model serving, *Prometheus* in conjunction with *Grafana* for monitoring, and *DVC* for data versioning. The architecture was largely confirmed to be effective during focus group validation. Only the model serving component was replaced with *BentoML* to facilitate scaling and load balancing in the following revision. The validation confirmed the findings of existing publications in the sense that constructing an MLOps architecture in an organization is a stepwise process. The prioritization of components must consider both the existing capabilities within the company and the value of the respective component for the individual use case. At the same time, the complexity of the IT infrastructure increases with each additional component, so it is sometimes better not to implement all components. Additionally, the architecture implementation has confirmed that to exploit the full potential of MLOps, a sophisticated DevOps infrastructure is required as the foundation. For example, the data and training pipeline tools are of limited use for deployment tasks.

The literature on the challenges of the productive use of machine learning is often based on reports from individual users or empirical studies of groups from individual companies. This work supports the academic field by synthesizing these publications and thus enabling more targeted research. Furthermore, the described best practices assist researchers in gaining a deeper understanding of the workflows in organizations. This is important because it facilitates the incorporation of findings from the industry into research and contributes to making research more relevant to real-world applications. The latter addresses the general concern Garousi et al. (2020) raised regarding the practical relevance of SE research. The final part of this research aimed to counter this criticism by collaborating with a company and underpinning the theoretical MLOps architecture with OSS components. This enabled the definition of an MLOps process that is not purely theoretical due to validation by experts from the industry.

In a practical context, this work enables organizations to proactively prepare for challenges that may arise with the productive use of machine learning. Additionally, this thesis supports them in addressing the challenges by providing practical recommendations for establishing mature machine learning processes. This is further supported by a comprehensive overview of OSS solutions that assist organizations in rolling out MLOps capabilities. Finally, while the designed MLOps architecture was validated in the context of an AI service provider, it still provides a starting point for adaptations to various contexts. Thus, the validation findings also represent valuable areas for reflection.

A central topic that has not been treated in depth within this thesis is online experimentation. While load balancing and scaling solutions for traditional microservices also work for machine learning serving, it is hard to realize sophisticated online experimentation methods like multi-armed bandit tests. In particular, how organizations can enable such techniques while maintaining control over their data and models is a topic that requires additional attention.

Furthermore, the agility and pace of the workflows gained with MLOps practices amplify the fundamental challenges associated with machine learning. One of these is the issue of explainability and fairness, which is being intensively studied in the academic community but is still not fully solved.[280] Another less actively addressed but still important topic is Machine Learning Security Operations (MLSecOps) which deals with mitigating security risks related to machine learning models, pipelines, and services.

Additionally, data versioning is still in a state of immaturity; here, sophisticated techniques are lacking that allow incremental versioning of various file formats in git-style.

[280] See, e.g., Awasthi et al. (2021); Klaise et al. (2020); Manerba and Guidotti (2021).

A1 Additional Data-Science Process Models

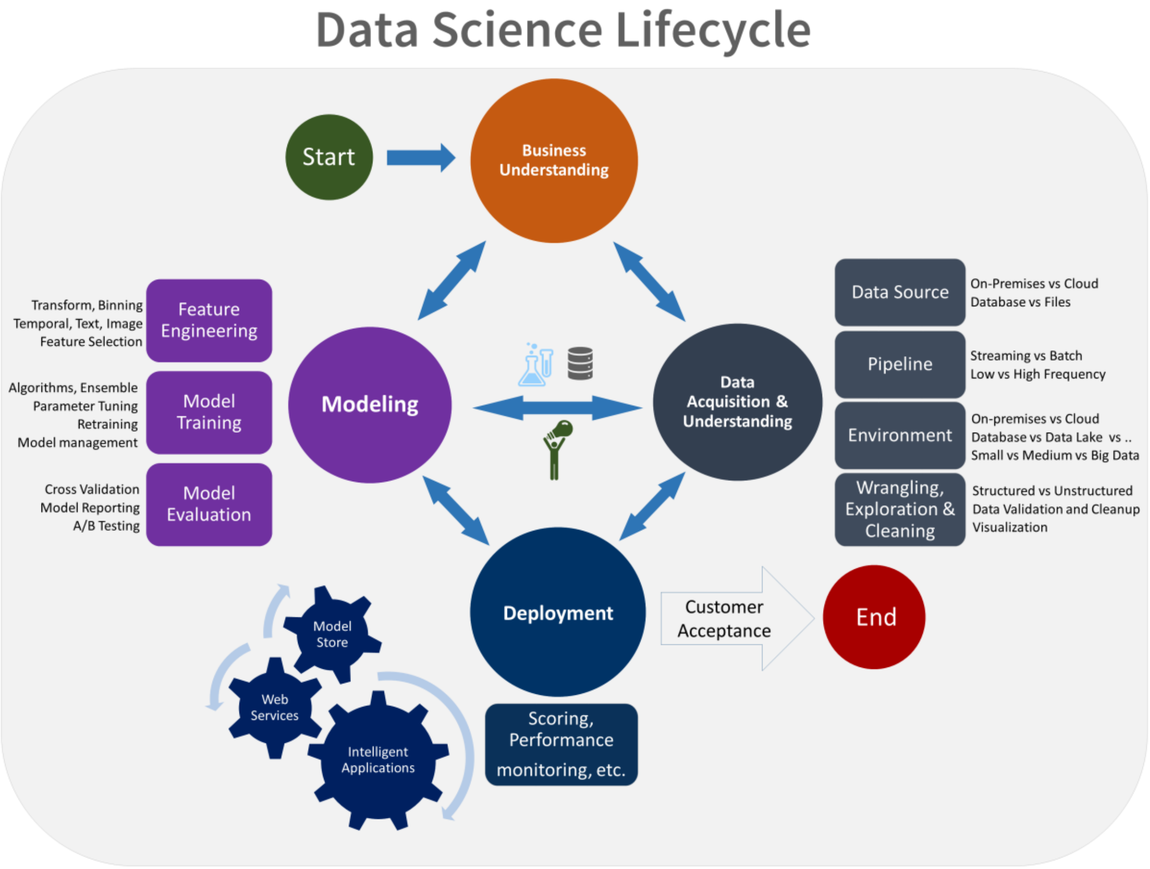


Figure 20: TDSP by Microsoft (2017)

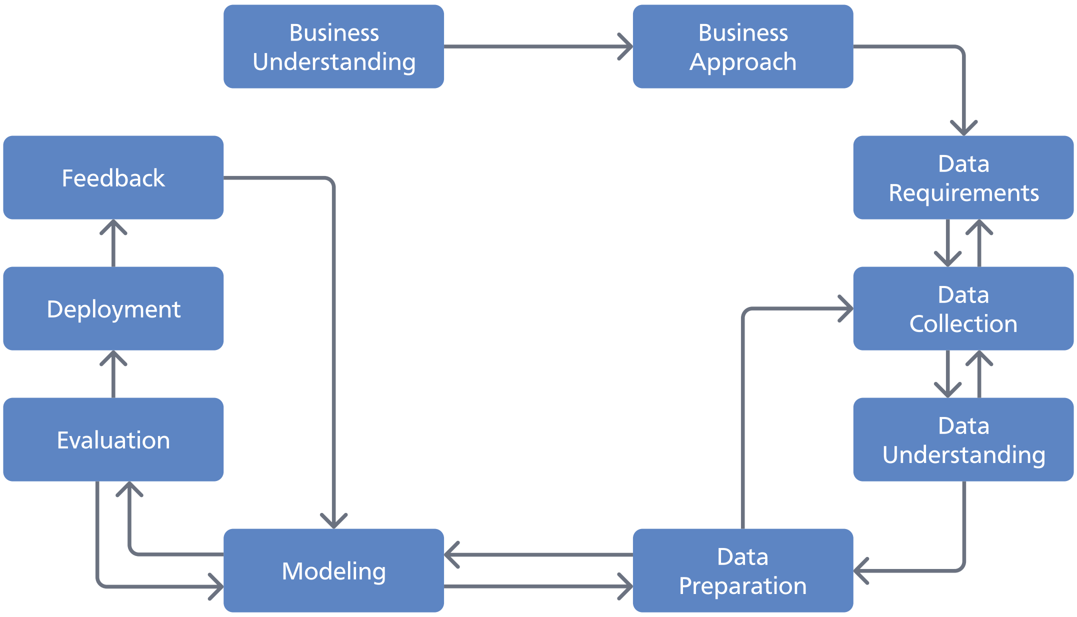


Figure 21: FMDS by IBM (2015)

A2 Multivocal Literature Review

Table 12: Guidelines for Conducting MLRs by Garousi et al. (2019).

#	Guideline
1	The provided typical process of an MLR can be applied to structure a protocol on how the review will be conducted. Alternatively, the standard protocol structure of SLR in SE can be applied and the provided guidelines can be considered as variation points.
2	Identify any existing reviews and plan/execute the MLR to explicitly provide usefulness for its intended audience (researchers and/or practitioners).
3	The decision whether to include the grey literature in a review study and to conduct an MLR study (instead of a conventional SLR) should be made systematically using a well-defined set of criteria/questions.
4	Based on your research goal and target audience, define the research (or "review") questions (RQs) in a way to (1) clearly relate to and systematically address the review goal, (2) match specific needs of the target audience, and (3) be as objective and measurable as possible.
5	Try adopting various RQ types but be aware that primary studies may not allow all question types to be answered.
6	Identify the relevant grey literature types and/or grey literature producers (data sources) for your review study early on.
7	General web search engines, specialized databases and websites, backlinks, and contacting individuals directly are ways to search for grey literature.
8	When searching for grey literature on SE topics, three possible stopping criteria for grey literature searches are: (1) Theoretical saturation, i.e., when no new concepts emerge from the search results; (2) Effort bounded, i.e., only include the top N search engine hits, and (3) Evidence exhaustion, i.e., extract all the evidence.
9	Combine inclusion and exclusion criteria for grey literature with quality assessment criteria.
10	In the source selection process of an MLR, one should ensure a coordinated integration of the source selection processes for grey literature and formal literature.
11	Apply and adapt the criteria authority of the producer, methodology, objectivity, date, novelty, impact, as well as outlet control, for study quality assessment of grey literature. ▪ Consider which criteria can already be applied for source selection ▪ There is no one-size-fits-all quality model for all types of grey literature. Thus, one should make suitable adjustments to the quality criteria checklist and consider reductions or extensions if focusing on particular studies such as survey, case study or experiment.
12	During the data extraction, systematic procedures and logistics, e.g., explicit "traceability" links between the extracted data and primary sources, should be utilized. Also, researchers should extract and record as much quantitative/qualitative data as needed to sufficiently address each RQ, to be used in the synthesis phase.
13	A suitable data synthesis method should be selected. Many grey literature sources are suitable for qualitative coding and synthesis. Some grey literature sources allow combination of survey results but lack of reporting rigor limits the meta-analysis. Quantitative analysis is possible on grey literature databases such as StackOverflow. Also argumentation theory can be beneficial for data synthesis from grey literature. Finally, the limitations of grey literature sources w.r.t. their evidence depth of experiment prevent meta-analysis.
14	The writing style of an MLR paper should match its target audience, i.e., researchers and/or practitioners.

Table 13: Inclusion Criteria of Grey Literature in SE Reviews by Garousi et al. (2019).[281]

#	Question	MLOps
1	Is the subject "complex" and not solvable by considering only the formal literature?	Yes
2	Is there a lack of volume or quality of evidence, or a lack of consensus of outcome measurement in the formal literature?	Yes
3	Is the contextual information important to the subject under study?	Yes
4	Is it the goal to validate or corroborate scientific outcomes with practical experiences?	No
5	Is it the goal to challenge assumptions or falsify results from practice using academic research and vice versa?	No
6	Would a synthesis of insights and evidence from the industrial and academic community be useful to one or even both communities?	Yes
7	Is there a large volume of practitioner sources indicating high practitioner interest in a topic?	Yes

[281] One or more positive answers suggest the inclusion of grey literature.

Table 14: Quality Assessment Checklist of Grey Literature for SE by Garousi et al. (2019)

Criteria	
Authority of the producer	▪ Is the publishing organization reputable? ▪ Is an individual author associated with a reputable organization? ▪ Has the author published other work in the field? ▪ Does the author have expertise in the area?
Methodology	▪ Does the source have a clearly stated aim? ▪ Does the source have a stated methodology? ▪ Is the source supported by authoritative, contemporary references? ▪ Are any limits clearly stated? ▪ Does the work cover a specific question? ▪ Does the work refer to a particular population or case?
Objectivity	▪ Does the work seem to be balanced in presentation? ▪ Is the statement in the sources as objective as possible? ▪ Is there vested interest? ▪ Are the conclusions supported by the data?
Date	▪ Does the item have a clearly stated date?
Position w.r.t. related sources	▪ Have key related grey literature or formal sources been linked to / discussed?
Novelty	▪ Does it enrich or add something unique to the research? ▪ Does it strengthen or refute a current position?
Impact	▪ Normalize all the following impact metrics into a single aggregated impact metric (when data are available): Number of citations, Number of backlinks, Number of social media shares, Number of comments posted for a specific online entry like a blog post or a video, Number of page or paper views
Outlet type	▪ 1st tier grey literature (High outlet control / High credibility, measure = 1): Books, magazines, theses, government reports, white papers, [pre-prints][282] ▪ 2nd tier grey literature (Moderate outlet control / Moderate credibility, measure = 0.5): Annual reports, news articles, presentations, videos, Q/A sites (such as StackOverflow), Wiki articles ▪ 3rd tier grey literature (Low outlet control / Low credibility, measure = 0): Blogs, emails, tweets

Table 15: MLR Inclusion and Exclusion Criteria

Inclusion criteria	Exclusion criteria
▪ Published in English or German ▪ Published 2017 or later ▪ Quality Assessment Score above 0.5	▪ Publications that refer to other semantic meanings of the search terms ▪ Publications concerning hardware challenges ▪ Publications that are only applicable to closed-source solutions

[282] Supplemented by author of this thesis.

A3 Search Protocol

Academic Database Search Engines: ACM Digital Library, arXiv, Google Scholar, IEEE Xplore, and Web of Science

Web Search Engine: Google Search

Search Keys:

- "Machine Learning" AND "Challenges" AND "Production"
- "MLOps" AND ("Components" OR "Patterns" OR "Best Practices")

Table 16: MLR Search Result Count After Filtering

Filtering Process	# Search Results
Unfiltered (Unique Results)	> 500.000
After Filtering based on the Title	75
After Filtering based on the Abstract	61
After Filtering based on the Content	52
After Quality Control Gate	50
After Snowballing	74

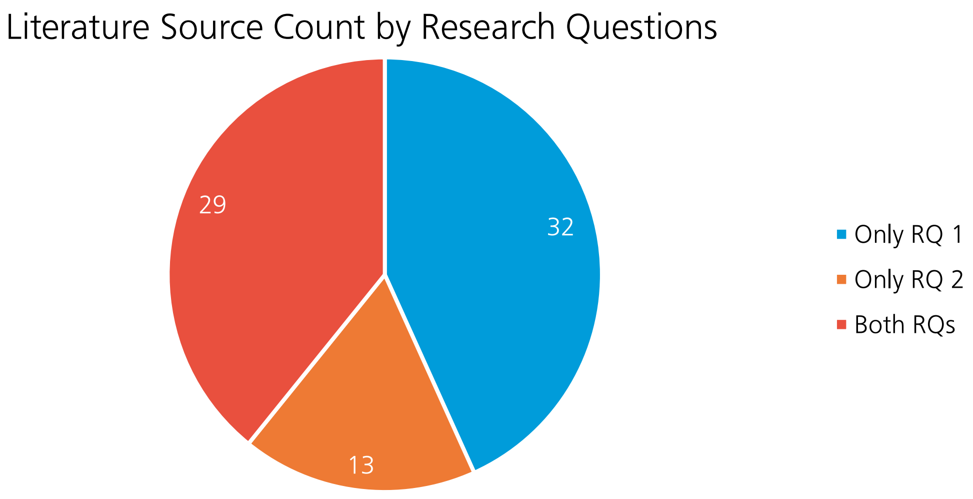


Figure 22: Literature Source Count by RQs

Table 17: Sources for Challenges in the Adoption of Machine Learning

Challenge	Sources
Lack of Data	Amershi et al. (2019); Baier et al. (2019a); Boutaba et al. (2018); Charrington (2020); Hazelwood et al. (2018); Heizmann et al. (2022); Ishikawa and Yoshioka (2019); John et al. (2020); Kocheturov et al. (2019); Lwakatare et al. (2020); Makinen et al. (2021); Morais (2021); Nahar et al. (2022); Paleyes et al. (2020); Polyzotis et al. (2017); Roh et al. (2021); Stonebraker and Rezig (2019); Walcott and Ali (2021); Whang and Lee (2020); Zhang et al. (2019); Zinkevich (2021)
Data Quality	Amershi et al. (2019); Baier et al. (2019a); Charrington (2020); Hazelwood et al. (2018); John et al. (2020), 2022); Kocheturov et al. (2019); Lakshmanan et al. (2020); Lwakatare et al. (2020); Makinen et al. (2021); Morais (2021); Nahar et al. (2022); Paleyes et al. (2020); Polyzotis et al. (2017); Polyzotis et al. (2019); Renggli et al. (2021); Roh et al. (2021); Stonebraker and Rezig (2019); Whang and Lee (2020)
Model Verification	Amershi et al. (2019); Arpteg et al. (2018); Boutaba et al. (2018); Habibullah and Horkoff (2021); Ishikawa and Yoshioka (2019); John et al. (2020); Kumeno (2020); Lakshmanan et al. (2020); Lwakatare et al. (2019); Morais (2021); Nahar et al. (2022); Nascimento et al. (2019); Paleyes et al. (2020); Schelter et al. (2018); Stonebraker and Rezig (2019); Walcott and Ali (2021); Zinkevich (2021)
Resource Constraints	Algorithmia (2021); Amershi et al. (2019); Arpteg et al. (2018); Charrington (2020); Hazelwood et al. (2018); Heizmann et al. (2022); John et al. (2020), 2022); Lakshmanan et al. (2020); Makinen et al. (2021); Morais (2021); Paleyes et al. (2020); Walcott and Ali (2021); Zhang et al. (2019)
Data Labeling	Baier et al. (2019b); Boutaba et al. (2018); Budd et al. (2021); Charrington (2020); John et al. (2020), 2022); Lwakatare et al. (2019); Lwakatare et al. (2020); Morais (2021); Paleyes et al. (2020); Roh et al. (2021); Schelter et al. (2018); Whang and Lee (2020)
Reproducibility	Algorithmia (2021); Amershi et al. (2019); Arpteg et al. (2018); Baier et al. (2019a); Habibullah and Horkoff (2021); Lakshmanan et al. (2020); Lwakatare et al. (2019); Makinen et al. (2021); Salama et al. (2021); Sato et al. (2019); Schelter et al. (2018); Schelter et al. (2017); Zaharia et al. (2018)
Data Dispersion	Charrington (2020); Dong and Rekatsinas (2018); Goes (2021); Lwakatare et al. (2019); Lwakatare et al. (2020); Makinen et al. (2021); Paleyes et al. (2020); Salama et al. (2021); Schelter et al. (2018); Whang and Lee (2020)
High Dimensional Data	Amershi et al. (2019); Baier et al. (2019a); Cai et al. (2018); Ferguson (2018); Kocheturov et al. (2019); Morais (2021)
Concept Drift	Algorithmia (2021); Baier et al. (2019a); Baier et al. (2019b); Charrington (2020); Derakhshan et al. (2019); Granlund et al. (2021); Hammami et al. (2017); Hazelwood et al. (2018); Heuvel and Tamburri (2020); John et al. (2020), 2022); Klaise et al. (2020); Kumeno (2020); Lakshmanan et al. (2020); Lu et al. (2018); Lwakatare et al. (2019); Lwakatare et al. (2020); Makinen et al. (2021); Martel et al. (2021); Nahar et al. (2022); Paleyes et al. (2020); Polyzotis et al. (2019); Raj (2021); Renggli et al. (2021); Salama et al. (2021); Xin et al. (2021); Zaharia et al. (2018)
Training-Serving Skews	Amershi et al. (2019); Boutaba et al. (2018); Ishikawa and Yoshioka (2019); John et al. (2020), 2022); Klaise et al. (2020); Lakshmanan et al. (2020); Lwakatare et al. (2019); Lwakatare et al. (2020); Martel et al. (2021); Polyzotis et al. (2019); Salama et al. (2021); Schelter et al. (2018); Zinkevich (2021)

Adversarial Attacks	Algorithmia (2021); Balakrishnan et al. (2020); Boutaba et al. (2018); Chui et al. (2021); Correia-Silva et al. (2018); Orekondy et al. (2019); Pal et al. (2019); Paleyes et al. (2020); Salama et al. (2021); Schelter et al. (2018); Yuan et al. (2022)
Lack of Standards	Algorithmia (2021); Baier et al. (2019a); Hazelwood et al. (2018); Heuvel and Tamburri (2020); Martel et al. (2021); Paleyes et al. (2020); Sato et al. (2019); Schelter et al. (2018); Stonebraker and Rezig (2019); Zaharia et al. (2018)
Heterogeneous Skill Levels	Algorithmia (2021); Amershi et al. (2019); Baier et al. (2019a); Granlund et al. (2021); Heuvel and Tamburri (2020); Ishikawa and Yoshioka (2019); John et al. (2020), 2022); Kumeno (2020); Makinen et al. (2021); Nahar et al. (2022); Paleyes et al. (2020); Salama et al. (2021); Sato et al. (2019); Schelter et al. (2018); Walcott and Ali (2021); Zaharia et al. (2018)
Model Fairness	Baier et al. (2019a); Habibullah and Horkoff (2021); Kumeno (2020); Lakshmanan et al. (2020); Morais (2021); Paleyes et al. (2020); Raj (2021); Salama et al. (2021); Stonebraker and Rezig (2019); Tamburri (2020); Whang and Lee (2020); Zhou (2018)
Regulatory Compliance	Arpteg et al. (2018); Baier et al. (2019a); Balakrishnan et al. (2020); Chui et al. (2021); Granlund et al. (2021); Habibullah and Horkoff (2021); Kumeno (2020); Lakshmanan et al. (2020); Makinen et al. (2021); Malle et al. (2017); Paleyes et al. (2020); Tamburri (2020)
Trust of End-Users	Algorithmia (2021); Arpteg et al. (2018); Baier et al. (2019a); Balakrishnan et al. (2020); Chui et al. (2021); Heizmann et al. (2022); John et al. (2020), 2022); Klaise et al. (2020); Paleyes et al. (2020); Zhou (2018)
Lack of Talent	Amershi et al. (2019); John et al. (2020), 2022); Makinen et al. (2021); Paleyes et al. (2020); Thieullent et al. (2020); Walcott and Ali (2021)

Table 18: Sources for Best Practices

Best Practice	Sources
Model Monitoring	Algorithmia (2021); Baier et al. (2019a); Balakrishnan et al. (2020); Charrington (2020); Chui et al. (2021); Derakhshan et al. (2019); Garg et al. (2021); Goes (2021); Granlund et al. (2021); Hewage and Meedeniya (2022); Karamitsos et al. (2020); Klaise et al. (2020); Lakshmanan et al. (2020); Lwakatare et al. (2020); Muralidhar et al. (2021); Nahar et al. (2022); Paleyes et al. (2020); Raj (2021); Ruf et al. (2021); Salama et al. (2021); Sato et al. (2019); Schelter et al. (2018); Serban et al. (2020); Symeonidis et al. (2022); Tagliabue (2021); Thieullent et al. (2020); Warnett and Zdun (2022); Zinkevich (2021)
Model Training Pipelines	Amershi et al. (2019); Balakrishnan et al. (2020); Charrington (2020); Chui et al. (2021); Derakhshan et al. (2019); Garg et al. (2021); Goes (2021); Granlund et al. (2021); Hazelwood et al. (2018); Lakshmanan et al. (2020); Muralidhar et al. (2021); O'Leary and Uchida (2020); Pölöskei (2022); Polyzotis et al. (2017); Raj (2021); Ruf et al. (2021); Salama et al. (2021); Sato et al. (2019); Schelter et al. (2018); Serban et al. (2020); Symeonidis et al. (2022); Tagliabue (2021); Warnett and Zdun (2022); Xin et al. (2021); Zinkevich (2021)
Model Registry	Amershi et al. (2019); Baier et al. (2019a); Charrington (2020); Chui et al. (2021); Garg et al. (2021); Goes (2021); Hewage and Meedeniya (2022); Karamitsos et al. (2020); Lakshmanan et al. (2020); Muralidhar et al. (2021); O'Leary and Uchida (2020); Raj (2021); Ruf et al. (2021); Salama et al. (2021); Schelter et al. (2017); Symeonidis et al. (2022); Warnett and Zdun (2022); Xin et al. (2021); Zaharia et al. (2018)
Data Pipelines	Amershi et al. (2019); Balakrishnan et al. (2020); Charrington (2020); Goes (2021); Granlund et al. (2021); Hewage and Meedeniya (2022); Lakshmanan et al. (2020); Muralidhar et al. (2021); Pölöskei (2022); Raj (2021); Ruf et al. (2021); Salama et al. (2021); Sato et al. (2019); Serban et al. (2020); Symeonidis et al. (2022); Tagliabue (2021); Warnett and Zdun (2022); Xin et al. (2021)
Deployment Pipelines	Amershi et al. (2019); Castro-Lopez and Vega-Lopez (2019); Garg et al. (2021); Goes (2021); Granlund et al. (2021); Hewage and Meedeniya (2022); Karamitsos et al. (2020); Lakshmanan et al. (2020); Muralidhar et al. (2021); Paleyes et al. (2020); Pölöskei (2022); Raj (2021); Ruf et al. (2021); Sato et al. (2019); Serban et al. (2020); Symeonidis et al. (2022); Warnett and Zdun (2022); Xin et al. (2021)
Data Monitoring	Baier et al. (2019a); Charrington (2020); Chui et al. (2021); Klaise et al. (2020); Lwakatare et al. (2020); Makinen et al. (2021); Nahar et al. (2022); Polyzotis et al. (2019); Raj (2021); Ruf et al. (2021); Salama et al. (2021); Sato et al. (2019); Schelter et al. (2018); Serban et al. (2020); Symeonidis et al. (2022); Warnett and Zdun (2022)
Feature Store	Amershi et al. (2019); Balakrishnan et al. (2020); Charrington (2020); Chui et al. (2021); Derakhshan et al. (2019); Goes (2021); Granlund et al. (2021); Hazelwood et al. (2018); Karamitsos et al. (2020); Lakshmanan et al. (2020); Raj (2021); Salama et al. (2021); Sato et al. (2019); Serban et al. (2020); Tagliabue (2021); Thieullent et al. (2020)
Experiment Tracking	Amershi et al. (2019); Charrington (2020); Granlund et al. (2021); Hazelwood et al. (2018); Lwakatare et al. (2020); Muralidhar et al. (2021); Raj (2021); Salama et al. (2021); Sato et al. (2019); Serban et al. (2020); Stonebraker and Rezig (2019); Symeonidis et al. (2022); Warnett and Zdun (2022); Zaharia et al. (2018)

Serving Strategy	Bac et al. (2022); Baier et al. (2019a); Charrington (2020); Derakhshan et al. (2019); Hazelwood et al. (2018); Karamitsos et al. (2020); Lakshmanan et al. (2020); Nahar et al. (2022); Pölöskei (2022); Ruf et al. (2021); Salama et al. (2021); Sato et al. (2019); Tagliabue (2021); Xin et al. (2021)
Data Versioning	Amershi et al. (2019); Granlund et al. (2021); Hewage and Meedeniya (2022); Karamitsos et al. (2020); Raj (2021); Ruf et al. (2021); Salama et al. (2021); Sato et al. (2019); Serban et al. (2020); Symeonidis et al. (2022); Warnett and Zdun (2022)
Code Versioning	Amershi et al. (2019); Garg et al. (2021); Goes (2021); Hewage and Meedeniya (2022); Karamitsos et al. (2020); Ruf et al. (2021); Salama et al. (2021); Sato et al. (2019)
Online Experimentation	Karamitsos et al. (2020); Lwakatare et al. (2020); Polyzotis et al. (2019); Salama et al. (2021); Sato et al. (2019); Schelter et al. (2018); Serban et al. (2020); Warnett and Zdun (2022)
Labeling Processes	Balakrishnan et al. (2020); Chui et al. (2021); Roh et al. (2021); Ruf et al. (2021); Serban et al. (2020); Symeonidis et al. (2022); Warnett and Zdun (2022)
Data Visualization	Raj (2021); Ruf et al. (2021); Salama et al. (2021); Sato et al. (2019); Tagliabue (2021); Warnett and Zdun (2022)

A4 Focus Group

Table 19: Focus Group Participants

Job Title	Professional Background[283]	SE Experience	Machine Learning Experience
Product Lead Platform	M.Sc. & Ph.D. in Business Informatics, Multiple years' experience in software development and product management for data-driven products	4-5 years	4-5 years
Product Owner	I finished my master's in business informatics end of last year and started being a product owner as my first job in the industry.	< 1 year	1-3 years
Data Scientist	Machine learning and computer vision engineer by *<COMPANY A>*, applying AI technique on satellite images. Data Scientist by *<COMPANY B>*, working on NLP and timeseries analysis tasks.	6-10 years	6-10 years
Data Scientist	I have worked as a pure Java developer for 4 years in India. Currently I am working as Data Scientist since 3 years. Here my task is to develop and improve AI related features.	6-10 years	1-3 years
Data Scientist	My professional experience is mostly with AI and related technologies especially in the field of NLP. In my previous roles, I was mostly involved with developing data models and big data processing pipelines for satellite data and retail e-commerce data.	4-5 years	1-3 years
Platform Environment Engineer	I support the AI team to developer their AI application and bringing it into production on our Nomad cluster infrastructure. In general, I now support all *<COMPANY B>* teams into building their CI/CD pipelines and help them in removing bottlenecks.	> 10 years	1-3 years
Development Lead	Started professionally with software development in 2007 to fund my studies. Did a PhD in natural language processing afterwards. Now working in the industry for about two years.	> 10 years	4-5 years
Software Engineer	Focused on formal methods in Master's. Then was doing system integration, testing, and deployment of a mission control system. Now working on a business application leveraging ML methods for planning cost, sales, revenue etc.	4-5 years	1-3 years

[283] The background information was written by the participants themselves, but the company names were removed by the author of this thesis.

Table 20: Focus Group Questions

#	Topic	Question
1	Feasibility	Are there design flaws that make it impossible to use the design as presented?
2	Missing Components	Are there components missing that are necessary for the deployment of an efficient and useful MLOps solution?
3	Prioritization	If you were allowed to use only one component from the design, which would it be?
4	Prioritization	If you would need to choose one component of this design to eliminate, what would you choose and why?
5	Pipeline Tool	What do you like best about dagster?
6	Pipeline Tool	What functionalities are you missing in dagster or where are you concerned?
7	Model Registry	What are the core improvements you see from using MLflow?
8	Model Registry	What functionalities are you missing MLflow or where are you concerned?
9	Metric Store	Can all the necessary monitoring parameters be calculated from the metrics that can be stored in Prometheus (counter, gauge, histogram, summary)?
10	Model Serving	Are GPUs relevant for AI applications in your daily work? (Serving/Training?)
11	Deepening	Is there a specific topic we want to circle back to from this discussion to add or expand on?
12	Missing Points	Is there anything else you want to add to the conversation about MLOps?

Table 21: Features of Conversations by Krueger and Casey (2015)

Characteristic	Description	Analysis Implication
Spontaneous comments	Participants do not anticipate the questions, and responses are not premeditated. Responses may not be carefully organized and logically presented.	Opinions may not be fully formed and are subject to change. Watch for modifications later in the discussion.
Inconsistent comments	Participants offer different opinions during the course of the focus group	Opinions may be changing, or new insights may result in different points of view. Or if conversation is excessively abstract, the differing points may be the result of ambiguous conversation.
Specific versus general comments	Often in focus groups the comments relate to a specific event or situation and may not be intended to be generalized.	Follow-up questions might be needed to clarify whether a comment can be generalized to other situations.
People change their minds	Participants might persuade others to a particular point of view	Listen for changes in opinion. Ask participants if they have changed their views and seek insights about the cause.
Words used differently	Individuals vary in how they use words.	Listen for words describing important concepts. Are words used in similar ways? Clarify these ambiguous words while you are still in the group. When doing analysis, code similar comments with the same code.
People repeat comments	Occasionally a participant will feel that a point of view needs more emphasis and will repeat the concept several times. Or someone will remember an important point and present it later in the focus group.	Be alert to the difference between frequency and extensiveness. Names on the transcript can be helpful in these situations.
Conversations tend to wander	Conversations wander, go off track, and loop back to previous comments.	The moderator needs to maintain the focus without being too heavy handed. Occasional off-topic conversations can yield insights but also eat up precious time.
People present a view with intensity or emotion	Intense emotions in a focus group tend to influence other participants. This intensity is less prevalent and tends to be less influential in other forms of qualitative data.	Watch for indicators of intensity in body language and in comments. Attempt to get comments from participants on how they are feeling and how this influences their views. Examine the field notes for indicators of intensity or emotion.

A5 Open-Source Tools

Table 22: Open-Source Tools with GitHub Links

Name	GitHub Link (Accessed: 2022-06-15)
Accelerate	https://github.com/huggingface/accelerate
Aim	https://github.com/aimhubio/aim
Apache Airflow	https://github.com/apache/airflow
Apache Spark	https://github.com/apache/spark
Argo Workflows	https://github.com/argoproj/argo-workflows
BentoML	https://github.com/bentoml/BentoML
ClearML	https://github.com/allegroai/clearml
CML	https://github.com/iterative/cml
Cortex	https://github.com/cortexlabs/cortex
dagster	https://github.com/dagster-io/dagster
Dask	https://github.com/dask/dask
DVC	https://github.com/iterative/dvc
dbt	https://github.com/dbt-labs/dbt-core
DeepSpeed	https://github.com/microsoft/DeepSpeed
doccano	https://github.com/doccano/doccano
Dolt	https://github.com/dolthub/dolt
Evidently	https://github.com/evidentlyai/evidently
facets	https://github.com/PAIR-code/facets
FEAST	https://github.com/feast-dev/feast
Flyte	https://github.com/flyteorg/flyte
Git LFS	https://github.com/git-lfs/git-lfs
Grafana	https://github.com/grafana/grafana
Great Expectations	https://github.com/great-expectations/great_expectations

H2O-3	https://github.com/h2oai/h2o-3
Horovod	https://github.com/horovod/horovod
Hyperopt	https://github.com/hyperopt/hyperopt
Ivy	https://github.com/unifyai/ivy
JupyterHub	https://github.com/jupyter/notebook
Katib	https://github.com/kubeflow/katib
Kedro	https://github.com/kedro-org/kedro
KerasTuner	https://github.com/keras-team/keras-tuner
KServe	https://github.com/kserve/kserve
Kubeflow	https://github.com/kubeflow/kubeflow
Label Studio	https://github.com/heartexlabs/label-studio
lakeFS	https://github.com/treeverse/lakeFS
MetaFlow	https://github.com/Netflix/metaflow
MindsDB	https://github.com/mindsdb/mindsdb
MLflow	https://github.com/mlflow/mlflow
Nuclio	https://github.com/nuclio/nuclio
Optuna	https://github.com/optuna/optuna
Opyrator	https://github.com/ml-tooling/opyrator
Pachyderm	https://github.com/pachyderm/pachyderm
Petastorm	https://github.com/uber/petastorm
Ploomber	https://github.com/ploomber/ploomber
Polyaxon	https://github.com/polyaxon/polyaxon
Prefect	https://github.com/PrefectHQ/prefect
Prometheus	https://github.com/prometheus/prometheus
Quilt	https://github.com/quiltdata/quilt
Ray ML	https://github.com/ray-project/ray

River	https://github.com/online-ml/river
Rubrix	https://github.com/recognai/rubrix
Sacred	https://github.com/IDSIA/sacred
Scikit-Optimize	https://github.com/scikit-optimize/scikit-optimize
Seldon Core	https://github.com/SeldonIO/seldon-core
Superset	https://github.com/apache/superset
Talos	https://github.com/autonomio/talos
TensorBoard	https://github.com/tensorflow/tensorboard
TensorFlow TFX	https://github.com/tensorflow/tfx
TerminusDB	https://github.com/terminusdb/terminusdb
TorchServe	https://github.com/pytorch/serve
Triton Inference Server	https://github.com/triton-inference-server/server
whylogs	https://github.com/whylabs/whylogs
ZenML	https://github.com/zenml-io/zenml

References

Adams, Richard J.; Smart, Palie; Huff, Anne S. (2017): Shades of Grey. Guidelines for Working with the Grey Literature in Systematic Reviews for Management and Organizational Studies. In: International Journal of Management Reviews, 19 (4), pp. 432-454.

Ahmed, Bakhtiyar; Dannhauser, Thomas; Philip, Nada (2018): A Lean Design Thinking Methodology (LDTM) for Machine Learning and Modern Data Projects. In: 2018 10th Computer Science and Electronic Engineering Conference (CEEC). Conference Proceedings: 19-21 September 2018, University of Essex, UK. IEEE, Piscataway, NJ, pp. 11-14.

Alegion (2019): Artificial Intelligence and Machine Learning Projects Are Obstructed by Data Issues. Global Survey of Data Scientists, AI Experts and Stakeholders. https://content.alegion.com/dimensional-researchs-survey, Accessed: 2022-06-29.

Alexander, Ian F.; Beus-Dukic, Ljerka (2009): Discovering Requirements. How to Specify Products and Services. Wiley, Chichester.

Algorithmia (2020): Machine Learning in Production. A Roadmap for Success. https://info.algorithmia.com/machine-learning-roadmap?utm_medium=whitepaper&utm_source=2021-ml-trends&utm_campaign=IC-1901-Machine-Learning-Roadmap, Accessed: 2022-02-18.

Algorithmia (2021): 2021 Enterprise Trends in Machine Learning. https://info.algorithmia.com/hubfs/2020/Reports/2021-Trends-in-ML/Algorithmia_2021_enterprise_ML_trends.pdf?hsLang=en-us, Accessed: 2022-02-18.

Amazon Web Services (2016): Amazon Machine Learning. Developer Guide. https://docs.aws.amazon.com/machine-learning/latest/dg/machinelearning-dg.pdf#training-ml-models, Accessed: 2022-06-04.

Amershi, Saleema; Begel, Andrew; Bird, Christian; DeLine, Robert; Gall, Harald; Kamar, Ece et al. (2019): Software Engineering for Machine Learning. A Case Study. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice. ICSE-SEIP 2019: 25-31 May 2019, Montréal, Canada: Proceedings. IEEE, Piscataway, NJ, pp. 291-300.

Andersen-Gott, Morten; Ghinea, Gheorghita; Bygstad, Bendik (2012): Why Do Commercial Companies Contribute to Open Source Software? In: International Journal of Information Management, 32 (2), pp. 106-117.

Andonie, Răzvan (2019): Hyperparameter Optimization in Learning Systems. In: Journal of Membrane Computing, 1 (4), pp. 279-291.

Arpteg, Anders; Brinne, Bjorn; Crnkovic-Friis, Luka; Bosch, Jan (2018): Software Engineering Challenges of Deep Learning. In: Bures, Tomas and Angelis, Lefteris (Eds.): SEAA 2018. 44th Euromicro Conference on Software Engineering and Advanced Applications: 29-31 August 2018, Prague, Czech Republic: Proceedings. IEEE, Piscataway, NJ, pp. 50-59.

Awasthi, Pranjal; Beutel, Alex; Kleindessner, Matthäus; Morgenstern, Jamie; Wang, Xuezhi (2021): Evaluating Fairness of Machine Learning Models Under Uncertain and Incomplete Information. In: FAccT '21. Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. ACM, New York, NY, pp. 206-214.

Aziz, Norshakirah A. (2015): Role of Focus Group Discussion (FGD) in e-Business Research. In: Open Access Library, 2 (01).

Bac, Ta P.; Tran, Minh N.; Kim, YoungHan (2022): Serverless Computing Approach for Deploying Machine Learning Applications in Edge Layer. In: 2022 International Conference on Information Networking (ICOIN). IEEE, Piscataway, NJ, pp. 396-401.

Baier, Lucas; Jöhren, Fabian; Seebacher, Stefan (2019a): Challenges in the Deployment and Operation of Machine Learning in Practice. In: ECIS '19. Proceedings of the 27th European Conference on Information Systems (ECIS). Association for Information Systems, Stockholm, pp. 1-15.

Baier, Lucas; Kühl, Niklas; Satzger, Gerhard (2019b): How to Cope with Change? Preserving Validity of Predictive Services over Time. In: Bui, Tung X. (Ed.): HICSS '19. Proceedings of the 52nd Annual Hawaii International Conference on System Sciences. University of Hawaii, Honolulu, HI, pp. 1085-1094.

Balakrishnan, Tara; Chui, Michael; Hall, Bryce; Henke, Nicolaus (2020): Global Survey. The State of AI in 2020. McKinsey & Company. https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/global-survey-the-state-of-ai-in-2020, Accessed: 2022-03-22.

Balter, Ben (2015): Open Source License Usage on GitHub.com. https://github.blog/2015-03-09-open-source-license-usage-on-github-com, Accessed: 2022-06-05.

Beck, Kent; Beedle, Mike; Bennekum, Arie van; Cockburn, Alistair; Cunningham, Ward; Fowler, Martin et al. (2001): Principles Behind the Agile Manifesto. https://agilemanifesto.org/principles.html, Accessed: 2022-06-17.

Bengio, Yoshua (2009): Learning Deep Architectures for AI. Now Publishers, Boston, MA.

Bezos, Jeff P. (2017): 2016 Letter to Shareholders. https://www.aboutamazon.com/news/company-news/2016-letter-to-shareholders, Accessed: 2022-06-29.

Bloor, Michael; Frankland, Jane; Thomas, Michelle; Robson, Kate (2002): Focus Groups in Social Research. SAGE Publications, London.

Boutaba, Raouf; Salahuddin, Mohammad A.; Limam, Noura; Ayoubi, Sara; Shahriar, Nashid; Estrada-Solano, Felipe; Caicedo, Oscar M. (2018): A Comprehensive Survey on Machine Learning for Networking. Evolution, Applications and Research Opportunities. In: Journal of Internet Services and Applications, 9, Article 16.

Budd, Samuel; Robinson, Emma C.; Kainz, Bernhard (2021): A Survey on Active Learning and Human-In-The-Loop Deep Learning for Medical Image Analysis. In: Medical Image Analysis, 71, Article 102062.

Cai, Jie; Luo, Jiawei; Wang, Shulin; Yang, Sheng (2018): Feature Selection in Machine Learning. A New Perspective. In: Neurocomputing, 300, pp. 70-79.

Carey, Martha A. (1995): Comment. Concerns in the Analysis of Focus Group Data. In: Qualitative Health Research, 5 (4), pp. 487-495.

Castro-Lopez, Oscar; Vega-Lopez, Ines F. (2019): Multi-Target Compiler for the Deployment of Machine Learning Models. In: Kandemir, Mahmut, Jimborean, Alexandra and Moseley, Tipp (Eds.): CGO '19. Proceedings of the 2019 IEEE/ACM International Symposium on Code Generation and Optimization: February 16-20, 2019, Washington, DC, USA. IEEE, Piscataway, NJ, pp. 280-281.

Charrington, Sam (2020): Kubernetes for MLOps. Scaling Enterprise Machine Learning, Deep Learning, and AI. This Week in Machine Learning & AI. https://www.hpe.com/us/en/resources/solutions/kubernetes-mlops.html, Accessed: 2022-03-17.

Chui, Michael; Hall, Bryce; Singla, Alex; Sukharevsky, Alexander (2021): The State of AI in 2021. McKinsey & Company. https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/global-survey-the-state-of-ai-in-2021, Accessed: 2022-03-22.

Cios, Krzysztof J.; Kurgan, Lukasz A. (2004): Trends in Data Mining and Knowledge Discovery. In: Jain, Lakhmi C. and Wu, Xindong (Eds.): Advanced Techniques in Knowledge Discovery and Data Mining. 1. ed., Springer, London, pp. 1-26.

Correia-Silva, Jacson R.; Berriel, Rodrigo F.; Badue, Claudine; Souza, Alberto F. de; Oliveira-Santos, Thiago (2018): Copycat CNN. Stealing Knowledge by Persuading Confession with Random Non-Labeled Data. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, Piscataway, NJ, pp. 1-8.

Derakhshan, Behrouz; Mahdiraji, Alireza R.; Rabl, Tilmann; Markl, Volker (2019): Continuous Deployment of Machine Learning Pipelines. In: Herschel, Melanie, Galhardas, Helena, Reinwald, Berthold, Fundulaki, Irini, Binnig, Carsten and Kaoudi, Zoi (Eds.): Advances in Database Technology - EDBT 2019. 22nd International Conference on Extending Database Technology, Lisbon, Portugal, March 26-29, 2019, Proceedings. University of Konstanz, Konstanz, pp. 379-408.

Dong, Xin L.; Rekatsinas, Theodoros (2018): Data Integration and Machine Learning. A Natural Synergy. In: Das, Gautam, Jermaine, Christopher, Bernstein, Philip and Eldawy, Ahmed (Eds.): SIGMOD'18. Proceedings of the 2018 International Conference on Management of Data: June 10-15, 2018, Houston, TX, USA. ACM, New York, NY, pp. 1645-1650.

European Commisson (2020): On Artificial Intelligence. A European Approach to Excellence and Trust. https://ec.europa.eu/info/sites/default/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf, Accessed: 2022-05-29.

Fan, Wei; Bifet, Albert (2013): Mining Big Data. Current Status, and Forecast to the Future. In: ACM SIGKDD Explorations Newsletter, 14 (2), pp. 1-5.

Fayyad, Usama; Piatetsky-Shapiro, Gregory; Smyth, Padhraic (1996a): From Data Mining to Knowledge Discovery in Databases. In: AI Magazine, 17 (3), pp. 37-54.

Fayyad, Usama; Piatetsky-Shapiro, Gregory; Smyth, Padhraic (1996b): The KDD Process for Extracting Useful Knowledge from Volumes of Data. In: Communications of the ACM, 39 (11), pp. 27-34.

Ferguson, Andrew L. (2018): Machine Learning and Data Science in Soft Materials Engineering. In: Journal of Physics: Condensed Matter, 30 (4), Article 043002.

Fitzgerald, Brian; Stol, Klaas-Jan (2017): Continuous Software Engineering. A Roadmap and Agenda. In: Journal of Systems and Software, 123, pp. 176-189.

Gama, João; Žliobaitė, Indrė; Bifet, Albert; Pechenizkiy, Mykola; Bouchachia, Abdelhamid (2014): A Survey on Concept Drift Adaptation. In: ACM Computing Surveys, 46 (4), Article 44.

Garg, Satvik; Pundir, Pradyumn; Rathee, Geetanjali; Gupta, Pradeep K.; Garg, Somya; Ahlawat, Saransh (2021): On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models Using MLOps. In: 2021 IEEE Fourth

International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). IEEE, Piscataway, NJ, pp. 25-28.

Garousi, Vahid; Borg, Markus; Oivo, Markku (2020): Practical Relevance of Software Engineering Research. Synthesizing the Community's Voice. In: Empirical Software Engineering, 25 (3), pp. 1687-1754.

Garousi, Vahid; Felderer, Michael; Mäntylä, Mika V. (2019): Guidelines for Including Grey Literature and Conducting Multivocal Literature Reviews in Software Engineering. In: Information and Software Technology, 106, pp. 101-121.

Gartner (2020): Gartner Identifies the Top Strategic Technology Trends for 2021. Analysts Explore Top Industry Trends at Gartner IT Symposium/Xpo 2020 Americas, October 19-22. https://www.gartner.com/en/newsroom/press-releases/2020-10-19-gartner-identifies-the-top-strategic-technology-trends-for-2021, Accessed: 2022-06-30.

Gentemann, Lukas; Termer, Frank; Weber, Anja (2021): Open-Source-Monitor. Studienbericht 2021. https://www.bitkom.org/Presse/Presseinformation/Grosse-Mehrheit-der-Unternehmen-setzt-auf-Open-Source, Accessed: 2022-06-05.

Gibson, Marcus; Arnott, David (2007): The Use of Focus Groups in Design Science Research. In: ACIS 2007 Proceedings, 14.

Gift, Noah; Deza, Alfredo (2021): Practical MLOps. Operationalizing Machine Learning Models. 1. ed., O'Reilly Media, Sebastopol, CA.

Gilflores, Javier; Alonso, Cristina G. (1995): Using Focus Groups in Educational Research. In: Evaluation Review, 19 (1), pp. 84-101.

Goes, Maurits van der (2021): Scaling Enterprise Recommender Systems for Decentralization. In: RecSys '21. Fifteenth ACM Conference on Recommender Systems. ACM, New York, NY, pp. 592-594.

Granlund, Tuomas; Stirbu, Vlad; Mikkonen, Tommi (2021): Towards Regulatory-Compliant MLOps. Oravizio's Journey from a Machine Learning Experiment to a Deployed Certified Medical Product. In: SN Computer Science, 2, Article 342.

Gujarati, Arpan; Elnikety, Sameh; He, Yuxiong; McKinley, Kathryn S.; Brandenburg, Björn B. (2017): Swayam. Distributed Autoscaling to Meet SLAs of Machine Learning Inference Services with Resource Efficiency. In: Jayaram, K. R. and Gandhi, Anshul (Eds.): Middleware '17. Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference. ACM, New York, NY, pp. 109-120.

Gujjar, Praveen J.; Kumar, Naveen V. (2022): Demystifying MLOps for Continuous Delivery of the Product. In: Asian Journal of Advances in Research, 13 (3), pp. 19-23.

Habibullah, Khan M.; Horkoff, Jennifer (2021): Non-functional Requirements for Machine Learning. Understanding Current Use and Challenges in Industry. In: Moreira, Ana, Schneider, Kurt, Vierhauser, Michael and Huang, Jane (Eds.): 29th IEEE International Requirements Engineering Conference. RE 2021: September 20-24, 2021: Online Event: Proceedings. IEEE, Piscataway, NJ, pp. 13-23.

Hammami, Zeineb; Mouelhi, Wiem; Said, Lamjed B. (2017): On-Line Self-Adaptive Framework for Tailoring a Neural-Agent Learning Model Addressing Dynamic Real-Time Scheduling Problems. In: Journal of Manufacturing Systems, 45, pp. 97-108.

Hartmann, Sebastian (2021): Sample Declaration of Consent. https://www.intern.tu-darmstadt.de/media/dezernat_i/id_gremien_ordner/ethikkommission/formulare_2019/EK_Muster-Einverstaendniserklaerung-english2.docx, Accessed: 2022-05-25.

Hawkins, Douglas M. (1980): Identification of Outliers. Springer, Dordrecht.

Hazelwood, Kim; Bird, Sarah; Brooks, David; Chintala, Soumith; Diril, Utku; Dzhulgakov, Dmytro et al. (2018): Applied Machine Learning at Facebook. A Datacenter Infrastructure Perspective. In: 24th IEEE International Symposium on High Performance Computer Architecture. Proceedings: 24-28 February 2018, Vienna, Austria. IEEE, Piscataway, NJ, pp. 620-629.

Heizmann, Michael; Braun, Alexander; Glitzner, Markus; Günther, Matthias; Hasna, Günther; Klüver, Christina et al. (2022): Implementing Machine Learning. Chances and Challenges. In: at - Automatisierungstechnik, 70 (1), pp. 90-101.

Heuvel, Willem-Jan van den; Tamburri, Damian A. (2020): Model-Driven ML-Ops for Intelligent Enterprise Applications. Vision, Approaches and Challenges. In: Shishkov, Boris (Ed.): Business Modeling and Software Design. 10th International Symposium, BMSD 2020, Berlin, Germany, July 6-8, 2020, Proceedings. Springer, Cham, pp. 169-181.

Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo; Ram, Sudha (2004): Design Science in Information Systems Research. In: MIS Quarterly, 28 (1), pp. 75-105.

Hewage, Nipuni; Meedeniya, Dulani (2022): Machine Learning Operations. A Survey on MLOps Tool Support. https://arxiv.org/pdf/2202.10169, Accessed: 2022-06-13.

Humble, Jez; Farley, David G. (2011): Continuous Delivery. Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, Upper Saddle River, NJ.

IBM (2015): Foundational Methodology for Data Science. https://tdwi.org/~/media/64511A895D86457E964174EDC5C4C7B1.PDF, Accessed: 2022-05-14.

Ishikawa, Fuyuki; Yoshioka, Nobukazu (2019): How Do Engineers Perceive Difficulties in Engineering of Machine-Learning Systems? Questionnaire Survey. In: 2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI 2019) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER & IP 2019). CESSER-IP 2019: 28 May 2019, Montréal, QC, Canada: Proceedings. IEEE, Piscataway, NJ, pp. 2-9.

Jabbari, Ramtin; Ali, Nauman bin; Petersen, Kai; Tanveer, Binish (2016): What is DevOps? In: Gregory, Peggy and Taylor, Katie (Eds.): XP '16 Workshops. Scientific Workshop Proceedings of XP2016. ACM, New York, NY, pp. 1-11.

Jati, Wikan K.; Lhaksmana, Kemas M. (2020): Optimization of Decision Tree Algorithm in Text Classification of Job Applicants Using Particle Swarm Optimization. In: 2020 3rd International Conference on Information and Communications Technology (ICOIACT). IEEE, Piscataway, NJ, pp. 201-205.

John, Meenu M.; Olsson, Helena H.; Bosch, Jan (2020): Developing ML/DL Models. A Design Framework. In: ICSSP '20. Proceedings of the International Conference on Software and System Processes. ACM, New York, NY, pp. 1-10.

John, Meenu M.; Olsson, Helena H.; Bosch, Jan (2022): Towards an AI-Driven Business Development Framework. A Multi-Case Study. In: Journal of Software: Evolution and Process, Article e2432.

Jorge, Alipio; Moyle, Steve; Voß, Angi (2002): Remote Collaborative Data Mining Through Online Knowledge Sharing. In: Camarinha-Matos, Luis M. (Ed.): Collaborative Business Ecosystems and Virtual Enterprises. IFIP TC5 / WG5.5 Third Working Conference on Infrastructures for Virtual Enterprises (PRO-VE'02) May 1-3, 2002, Sesimbra, Portugal. Springer, New York, NY, pp. 497-504.

Karamitsos, Ioannis; Albarhami, Saeed; Apostolopoulos, Charalampos (2020): Applying DevOps Practices of Continuous Automation for Machine Learning. In: Information, 11 (7), Article 363.

Karmaker, Shubhra K.; Hassan, Mahadi Md.; Smith, Micah J.; Xu, Lei; Zhai, Chengxiang; Veeramachaneni, Kalyan (2022): AutoML to Date and Beyond. Challenges and Opportunities. In: ACM Computing Surveys, 54 (8), Article 175.

Khurana, Udayan; Turaga, Deepak; Samulowitz, Horst; Parthasrathy, Srinivasan (2016): Cognito. Automated Feature Engineering for Supervised Learning. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). IEEE, Piscataway, NJ, pp. 1304-1307.

Kitchenham, Barbara A.; Budgen, David; Brereton, Pearl (2016): Evidence-Based Software Engineering and Systematic Reviews. CRC Press, Boca Raton, FL.

Kitzinger, Jenny (1994): The Methodology of Focus Groups. The Importance of Interaction Between Research Participants. In: Sociology of Health and Illness, 16 (1), pp. 103-121.

Klaise, Janis; Looveren, Arnaud van; Cox, Clive; Vacanti, Giovanni; Coca, Alexandru (2020): Monitoring and Explainability of Models in Production. Workshop on Challenges in Deploying and Monitoring Machine Learning Systems (ICML 2020). https://arxiv.org/pdf/2007.06299, Accessed: 2022-06-02.

Kocheturov, Anton; Pardalos, Panos M.; Karakitsiou, Athanasia (2019): Massive Datasets and Machine Learning for Computational Biomedicine. Trends and Challenges. In: Annals of Operations Research, 276 (1-2), pp. 5-34.

Krueger, Richard A.; Casey, Mary A. (2015): Focus Groups. A Practical Guide for Applied Research. 5. ed., SAGE Publications, Los Angeles, CA.

Kumeno, Fumihiro (2020): Sofware Engneering Challenges for Machine Learning Applications. A Literature Review. In: Intelligent Decision Technologies, 13 (4), pp. 463-476.

Lakshmanan, Valliappa; Robinson, Sara; Munn, Michael (2020): Machine Learning Design Patterns. Solutions to Common Challenges in Data Preparation, Model Building, and MLOps. 1. ed., O'Reilly Media, Sebastopol, CA.

Lau, Sam; Drosos, Ian; Markel, Julia M.; Guo, Philip J. (2020): The Design Space of Computational Notebooks. An Analysis of 60 Systems in Academia and Industry. In: 2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, Piscataway, NJ, pp. 1-11.

Lauesen, Soren (2002): Software Requirements. Styles and Techniques. 1. ed., Addison-Wesley, London.

Li, Zijun; Chen, Quan; Xue, Shuai; Ma, Tao; Yang, Yong; Song, Zhuo; Guo, Minyi (2020): Amoeba. QoS-Awareness and Reduced Resource Usage of Microservices with Serverless Computing. In: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, Piscataway, NJ, pp. 399-408.

Lloyd, Wes; Ramesh, Shruti; Chinthalapati, Swetha; Ly, Lan; Pallickara, Shrideep (2018): Serverless Computing. An Investigation of Factors Influencing Microservice Performance. In: 2018 IEEE International Conference on Cloud Engineering (IC2E). IEEE, Piscataway, NJ, pp. 159-169.

Lu, Jie; Liu, Anjin; Dong, Fan; Gu, Feng; Gama, João; Zhang, Guangquan (2018): Learning Under Concept Drift. A Review. In: IEEE Transactions on Knowledge and Data Engineering, 31 (12), pp. 2346-2363.

Lunt, Peter; Livingstone, Sonia (1996): Rethinking the Focus Group in Media and Communications Research. In: Journal of Communication, 46 (2), pp. 79-98.

Lwakatare, Lucy E.; Raj, Aiswarya; Bosch, Jan; Olsson, Helena H.; Crnkovic, Ivica (2019): A Taxonomy of Software Engineering Challenges for Machine Learning Systems. An Empirical Investigation. In: Kruchten, Philippe, Fraser, Steven and Coallier, François (Eds.): Agile Processes in Software Engineering and Extreme Programming. 20th International Conference, XP 2019, Montrèal, QC, Canada, May 21-25, 2019: Proceedings. Springer, Cham, pp. 227-243.

Lwakatare, Lucy E.; Raj, Aiswarya; Crnkovic, Ivica; Bosch, Jan; Olsson, Helena H. (2020): Large-Scale Machine Learning Systems in Real-World Industrial Settings. A Review of Challenges and Solutions. In: Information and Software Technology, 127, Article 106368.

Mainali, Kiran; Ehrlinger, Lisa; Matskin, Mihhail; Himmelbauer, Johannes (2021): Discovering DataOps. A Comprehensive Review of Definitions, Use Cases, and Tools. In: Bhulai, Sandjai (Ed.): Data Analytics 2021. The Tenth International Conference on Data Analytics: October 3-7, 2021, Barcelona, Spain. IARIA, Wilmington, DE, pp. 61-69.

Makinen, Sasu; Skogstrom, Henrik; Laaksonen, Eero; Mikkonen, Tommi (2021): Who Needs MLOps. What Data Scientists Seek to Accomplish and How Can MLOps Help? In: 2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI. WAIN 2021: Virtual (Originally Madrid, Spain), 22-30 May 2021: Proceedings. IEEE, Piscataway, NJ, pp. 109-112.

Malle, Bernd; Kieseberg, Peter; Holzinger, Andreas (2017): Do Not Disturb? Classifier Behavior on Perturbed Datasets. In: Holzinger, Andreas, Kieseberg, Peter, Tjoa, A M. and Weippl, Edgar R. (Eds.): Machine Learning and Knowledge Extraction. First IFIP TC 5, Wg 8.4, 8.9, 12.9 International Cross-Domain Conference, CD-Make 2017: Reggio, Italy, August 29-September 1, 2017: Proceedings. Bd. 10410, Springer, Cham, pp. 155-173.

Manerba, Marta M.; Guidotti, Riccardo (2021): FairShades. Fairness Auditing via Explainability in Abusive Language Detection Systems. In: 2021 IEEE Third International Conference on Cognitive Machine Intelligence (CogMI). IEEE, Piscataway, NJ, pp. 34-43.

Martel, Yannick; Roßmann, Arne; Sultanow, Eldar; Weiß, Oliver; Wissel, Matthias; Pelzel, Frank; Seßler, Matthias (2021): Software Architecture Best Practices for Enterprise Artificial Intelligence. In: Reussner, Ralf, Koziolek, Anne and Heinrich, Robert (Eds.): Informatik 2020. Fachtagung vom 28. September - 2. Oktober 2020. Gesellschaft für Informatik, Bonn, pp. 165-181.

Martinez-Plumed, Fernando; Contreras-Ochando, Lidia; Ferri, Cesar; Hernandez-Orallo, Jose; Kull, Meelis; Lachiche, Nicolas et al. (2021): CRISP-DM Twenty Years Later. From Data Mining Processes to Data Science Trajectories. In: IEEE Transactions on Knowledge and Data Engineering, 33 (8), pp. 3048-3061.

Merton, Robert K.; Kendall, Patricia L. (1946): The Focused Interview. In: American Journal of Sociology, 51 (6), pp. 541-557.

Microsoft (2017): What is the Team Data Science Process? https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview, Accessed: 2022-06-18.

Morais, Rui M. (2021): On the Suitability, Requisites, and Challenges of Machine Learning. In: Journal of Optical Communications and Networking, 13 (1), A1-A12.

Morgan, David L. (1996): Focus Groups. In: Annual Review of Sociology, 22 (1), pp. 129-152.

Munappy, Aiswarya R.; Mattos, David I.; Bosch, Jan; Olsson, Helena H.; Dakkak, Anas (2020): From Ad-Hoc Data Analytics to DataOps. In: ICSSP '20. Proceedings of the International Conference on Software and System Processes. ACM, New York, NY, pp. 165-174.

Muralidhar, Nikhil; Muthiah, Sathappah; Butler, Patrick; Jain, Manish; Yu, Yu; Burne, Katy et al. (2021): Using AntiPatterns to Avoid MLOps Mistakes. https://arxiv.org/pdf/2107.00079, Accessed: 2022-06-13.

Nahar, Nadia; Zhou, Shurui; Lewis, Grace; Kästner, Christian (2022): Collaboration Challenges in Building ML-Enabled Systems. Communication, Documentation, Engineering, and Process. https://arxiv.org/pdf/2110.10234, Accessed: 2022-06-03.

Nascimento, Elizamary d.S.; Ahmed, Iftekhar; Oliveira, Edson; Palheta, Marcio P.; Steinmacher, Igor; Conte, Tayana (2019): Understanding Development Process of Machine Learning Systems. Challenges and Solutions. In: 13th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM 2019: 19-20 September 2019, Porto de Galinhas, Pernambuco, Brazil: Proceedings. IEEE, Piscataway, NJ, pp. 1-6.

O'Leary, Katie; Uchida, Makoto (2020): Common Problems With Creating Machine Learning Pipelines From Existing Code. In: Third Conference on Machine Learning and Systems (MLSys). Workshop on MLOps Systems.

Open Source Initiative (2007): The Open Source Definition. https://opensource.org/docs/osd, Accessed: 2022-06-05.

Orekondy, Tribhuvanesh; Schiele, Bernt; Fritz, Mario (2019): Knockoff Nets. Stealing Functionality of Black-Box Models. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway, NJ, pp. 4949-4958.

Pal, Soham; Gupta, Yash; Shukla, Aditya; Kanade, Aditya; Shevade, Shirish; Ganapathy, Vinod (2019): A Framework for the Extraction of Deep Neural Networks by Leveraging Public Data. https://arxiv.org/abs/1905.09165, Accessed: 2022-07-01.

Paleyes, Andrei; Urma, Raoul-Gabriel; Lawrence, Neil D. (2020): Challenges in Deploying Machine Learning. A Survey of Case Studies. https://arxiv.org/pdf/2011.09926, Accessed: 2022-07-01.

Peffers, Ken; Tuunanen, Tuure; Rothenberger, Marcus A.; Chatterjee, Samir (2007): A Design Science Research Methodology for Information Systems Research. In: Journal of Management Information Systems, 24 (3), pp. 45-77.

Pölöskei, István (2022): MLOps Approach in the Cloud-Native Data Pipeline Design. In: Acta Technica Jaurinensis, 15 (1), pp. 1-6.

Polyzotis, Neoklis; Roy, Sudip; Whang, Steven E.; Zinkevich, Martin (2017): Data Management Challenges in Production Machine Learning. In: Chirkova, Rada (Ed.): SIGMOD '17. Proceedings of the 2017 ACM International Conference on Management of Data. ACM, New York, NY, pp. 1723-1726.

Polyzotis, Neoklis; Zinkevich, Martin; Roy, Sudip; Breck, Eric; Whang, Steven E. (2019): Data Validation for Machine Learning. In: Talwalkar, Ameet, Smith, Virginia and Zaharia, Matei (Eds.): Proceedings of Machine Learning and Systems (MLSys 2019), pp. 334-347.

Powell, Richard A.; Single, Helen M. (1996): Focus Groups. In: International Journal for Quality in Health Care, 8 (5), pp. 499-504.

Rabiee, Fatemeh (2004): Focus-Group Interview and Data Analysis. In: Proceedings of the Nutrition Society, 63 (4), pp. 655-660.

Raj, Emmanuel (2021): Engineering MLOps. Rapidly Build, Test, and Manage Production-Ready Machine Learning Life Cycles at Scale. Packt Publishing, Birmingham.

Renggli, Cedric; Rimanic, Luka; Gürel, Nezihe M.; Karlaš, Bojan; Wu, Wentao; Zhang, Ce (2021): A Data Quality-Driven View of MLOps. In: IEEE Data Engineering Bulletin, 44 (1), pp. 11-23.

Richardson, Christine A.; Rabiee, Fatemeh (2001): A Question of Access. An Exploration of the Factors that Influence the Health of Young Males Aged 15 to 19 Living in Corby and their Use of Health Care Services. In: Health Education Journal, 60 (1), pp. 3-16.

Roh, Yuji; Heo, Geon; Whang, Steven E. (2021): A Survey on Data Collection for Machine Learning. A Big Data - AI Integration Perspective. In: IEEE Transactions on Knowledge and Data Engineering, 33 (4), pp. 1328-1347.

Ruf, Philipp; Madan, Manav; Reich, Christoph; Ould-Abdeslam, Djaffar (2021): Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools. In: Applied Sciences, 11 (19), Article 8861.

Russell, Stuart J.; Norvig, Peter (2016): Artificial Intelligence. A Modern Approach. 3. ed., Pearson, Boston.

Salama, Khalid; Kazmierczak, Jarek; Schut, Donna (2021): Practitioners Guide to MLOps. A Framework for Continuous Delivery and Automation of Machine Learning. Google. https://services.google.com/fh/files/misc/practitioners_guide_to_mlops_whitepaper.pdf, Accessed: 2022-03-18.

Samuel, Arthur L. (1959): Some Studies in Machine Learning Using the Game of Checkers. In: IBM Journal, 3 (3), pp. 535-553.

Sato, Danilo; Wider, Arif; Windheuser, Christoph (2019): Continuous Delivery for Machine Learning. Automating the End-To-End Lifecycle of Machine Learning Applications. https://martinfowler.com/articles/cd4ml.html, Accessed: 2022-06-07.

Schelter, Sebastian; Bießmann, Felix; Januschowski, Tim; Salinas, David; Seufert, Stephan; Szarvas, Gyuri (2018): On Challenges in Machine Learning Model Management. In: IEEE Data Engineering Bulletin, 41 (4), pp. 5-15.

Schelter, Sebastian; Böse, Joos-Hendrik; Kirschnick, Johannes; Klein, Thoralf; Seufert, Stephan (2017): Automatically Tracking Metadata and Provenance of Machine Learning Experiments. In: Bird, Sarah, Crankshaw, Dan, Gibson, Garth, Gonzalez, Joseph, Lakshmiratan, Aparna, Li, Li E. et al. (Eds.): Proceedings of Conference on Neural Information Processing Systems (NIPS 2017). Machine Learning Systems Workshop. Curran Associates, Red Hook, NY, pp. 1-8.

Schoettle, Hendrik (2019): Open Source License Compliance. Why and How? In: Computer, 52 (8), pp. 63-67.

Sculley, D.; Holt, Gary; Golovin, Daniel; Davydov, Eugene; Phillips, Todd; Ebner, Dietmar et al. (2015): Hidden Technical Debt in Machine Learning Systems. In: Advances in Neural Information Processing Systems, 28.

Serban, Alex; Blom, Koen van der; Hoos, Holger; Visser, Joost (2020): Adoption and Effects of Software Engineering Best Practices in Machine Learning. In: ESEM '20. Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). ACM, New York, NY, pp. 1-12.

Shearer, Colin (2000): The CRISP-DM Model. The New Blueprint for Data Mining. In: Journal of Data Warehousing, 5 (4), pp. 13-22.

Stonebraker, Michael; Rezig, El K. (2019): Machine Learning and Big Data. What is Important? In: IEEE Data Engineering Bulletin, 42 (4), pp. 3-7.

Sutton, Steve G.; Arnold, Vicky (2013): Focus Group Methods. Using Interactive and Nominal Groups to Explore Emerging Technology-Driven Phenomena in Accounting and Information Systems. In: International Journal of Accounting Information Systems, 14 (2), pp. 81-88.

Symeonidis, Georgios; Nerantzis, Evangelos; Kazakis, Apostolos; Papakostas, George A. (2022): MLOps. Definitions, Tools and Challenges. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, Piscataway, NJ, pp. 453-460.

Tagliabue, Jacopo (2021): You Do Not Need a Bigger Boat. Recommendations at Reasonable Scale in a (Mostly) Serverless and Open Stack. In: RecSys '21. Fifteenth ACM Conference on Recommender Systems. ACM, New York, NY, pp. 598-600.

Tamburri, Damian A. (2020): Sustainable MLOps. Trends and Challenges. In: 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. IEEE, Piscataway, NJ, pp. 17-23.

The National New Generation Artificial Intelligence Governance Specialist Committee (2021): Ethical Norms for New Generation Artificial Intelligence Released. https://cset.georgetown.edu/wp-content/uploads/t0400_AI_ethical_norms_EN.pdf, Accessed: 2022-06-04.

Thieullent, Anne-Laure; Baerd, Marie-caroline; Yardi, Ashwin; Tolido, Ron; Schladitz, Fabian; Kurtz, Jerry et al. (2020): The AI-Powered Enterprise. Unlocking the Potential of AI at Scale. Capgemini Research Institute. https://www.capgemini.com/gb-en/research/the-ai-powered-enterprise/, Accessed: 2022-03-22.

Tremblay, Monica C.; Hevner, Alan R.; Berndt, Donald J. (2010): The Use of Focus Groups in Design Science Research. In: Hevner, Alan R., Chatterjee, Samir, Gray, Paul and Baldwin, Carliss Y. (Eds.): Design Research in Information Systems. 22, Springer, New York, NY, pp. 121-143.

Tyndall, Jess (2010): AACODS Checklist. Archived at the Flinders Academic Commons. https://dspace.flinders.edu.au/jspui/bitstream/2328/3326/4/AACODS_Checklist.pdf, Accessed: 2022-02-11.

Walcott, Terry H.; Ali, Maaruf (2021): Machine Learning for Smaller Firms. Challenges and Opportunities. In: Miraz, Mahdi H., Southall, Garfield, Ali, Maaruf, Ware, Andrew and Soomro, Safeeullah (Eds.): 2021 International Conference on Computing, Electronics & Communications Engineering (iCCECE). University of Essex, Southend Campus, UK: Held Online as a Live Interactive Virtual Conference, 16th-17th August, 2021: Proceedings. IEEE, Piscataway, NJ, pp. 82-86.

Warnett, Stephen J.; Zdun, Uwe (2022): Architectural Design Decisions for Machine Learning Deployment. In: 19th IEEE International Conference on Software Architecture (ICSA 2022). IEEE, Piscataway, NJ, pp. 90-100.

Weaver, John F. (2018): Regulation of Artificial Intelligence in the United States. In: Barfield, Woodrow and Pagallo, Ugo (Eds.): Research Handbook on the Law of Artificial Intelligence. Edward Elgar Publishing, Cheltenham, pp. 155-212.

Whang, Steven E.; Lee, Jae-Gil (2020): Data Collection and Quality Challenges for Deep Learning. In: Proceedings of the VLDB Endowment, 13 (12), pp. 3429-3432.

Wieringa, Roel J. (2014): Design Science Methodology for Information Systems and Software Engineering. Springer, Berlin.

Wohlin, Claes (2014): Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In: Shepperd, Martin (Ed.): EASE '14. Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, New York, NY, pp. 1-10.

Wohlin, Claes; Runeson, Per; Höst, Martin; Magnus, Ohlsson C.; Regnell, Björn; Wesslén, Anders (2012): Experimentation in Software Engineering. Springer, Berlin.

Xin, Doris; Miao, Hui; Parameswaran, Aditya; Polyzotis, Neoklis (2021): Production Machine Learning Pipelines. Empirical Analysis and Optimization Opportunities. In: Li, Guoliang (Ed.): SIGMOD '21. Proceedings of the 2021 International Conference on Management of Data. ACM, New York, NY, pp. 2639-2652.

Yuan, Xiaoyong; Ding, Leah; Zhang, Lan; Li, Xiaolin; Wu, Dapeng O. (2022): ES Attack. Model Stealing Against Deep Neural Networks Without Data Hurdles. In: IEEE Transactions on Emerging Topics in Computational Intelligence (forthcoming).

Zaharia, Matei; Chen, Andrew; Davidson, Aaron; Ghodsi, Ali; Hong, Sue A.; Konwinski, Andy et al. (2018): Accelerating the Machine Learning Lifecycle with MLflow. In: IEEE Data Engineering Bulletin, 41 (4), pp. 39-45.

Zhang, Yuxiang; Fu, Jiamei; Yang, Chengyi; Xiao, Chunjing (2019): A Local Expansion Propagation Algorithm for Social Link Identification. In: Knowledge and Information Systems, 60 (1), pp. 545-568.

Zheng, Alice; Casari, Amanda (2018): Feature Engineering for Machine Learning. Principles and Techniques for Data Scientists. 1. ed., O'Reilly Media, Beijing.

Zhou, Zhi-Hua (2018): Machine Learning Challenges and Impact. An Interview With Thomas Dietterich. In: National Science Review, 5 (1), pp. 54-58.

Zinkevich, Martin (2021): Rules of Machine Learning. Best Practices for ML Engineering. https://developers.google.com/machine-learning/guides/rules-of-ml, Accessed: 2022-04-09.