Automating machine learning: A meta-synthesis of MLOps tools, frameworks and architectures

Danylo O. Hanchuk[1], Serhiy O. Semerikov[1,2,3,4,5]

[1]*Kryvyi Rih State Pedagogical University, 54 Universytetskyi Ave., Kryvyi Rih, 50086, Ukraine*
[2]*Institute for Digitalisation of Education of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine*
[3]*Zhytomyr Polytechnic State University, 103 Chudnivsyka Str., Zhytomyr, 10005, Ukraine*
[4]*Kryvyi Rih National University, 11 Vitalii Matusevych Str., Kryvyi Rih, 50027, Ukraine*
[5]*Academy of Cognitive and Natural Sciences, 54 Universytetskyi Ave., Kryvyi Rih, 50086, Ukraine*

Abstract

Automating the end-to-end lifecycle of machine learning models is critical for their effective operationalization. Various tools, frameworks and architectures have emerged to support Machine Learning Operations (MLOps) practices. This paper presents a meta-synthesis of existing reviews to provide a comprehensive overview of such enabling technologies for MLOps. The capabilities and features offered by common commercial and open-source MLOps platforms are compared. Patterns in the MLOps architecture and design philosophies are identified. The role of containers, orchestration, configuration management, and infrastructure automation in ML pipelines is examined. Approaches for model deployment on cloud and edge are also discussed. The synthesis offers insights for tool selection and usage to automate enterprise-scale machine learning.

Keywords

MLOps, automation, tools, frameworks, architecture, model deployment, ML pipelines, meta-synthesis

1. Introduction

In the modern world, machine learning is becoming an increasingly important technology that finds application in various fields such as finance, healthcare, industry, retail, etc. [1] However, despite significant progress in the development of machine learning algorithms and models, their effective deployment in production environments remains a challenging task [2, 3]. This is due to a number of factors, such as the need to ensure scalability, reproducibility, security, and reliability of models, as well as the complexity of integrating development and operation processes.

To solve these problems, the MLOps (Machine Learning Operations) methodology has emerged, which aims to apply the principles and practices of DevOps to the development and deployment processes of machine learning models [4, 5]. MLOps covers a wide range of practices, such as automation of machine learning pipelines, versioning of data and models, monitoring model performance, experiment management, etc. [6, 7]. Research shows that the application of MLOps practices can significantly increase the efficiency and reliability of deploying machine learning models in production environments [8, 9].

At the same time, despite the significant interest in the topic of MLOps from both scientists and practitioners, there are still certain gaps and unresolved problems in this area. In particular, there are no generally accepted standards and best practices for implementing MLOps, issues of integrating MLOps with other approaches (DataOps, ModelOps, AIOps, etc.) are insufficiently researched, and there is a need to develop new tools and platforms to automate MLOps processes [5, 10, 11].

This work is aimed at solving the current problem of defining and analysing MLOps practices necessary for the effective deployment of machine learning models. The basis for performing the work

CS&SE@SW 2024: 7th Workshop for Young Scientists in Computer Science & Software Engineering, December 27, 2024, Kryvyi Rih, Ukraine
danilhanchuk@gmail.com (D. O. Hanchuk); semerikov@gmail.com (S. O. Semerikov)
https://acnsci.org/semerikov (S. O. Semerikov)
0009-0004-6474-3521 (D. O. Hanchuk); 0000-0003-0789-0272 (S. O. Semerikov)

is the need to systematize and generalize knowledge about MLOps practices, as well as the need to develop recommendations for their implementation in organizations to increase the efficiency and reliability of deploying machine learning models in production environments.

According to the aim, the following main objectives of the study are defined:

1. Perform a meta-synthesis of systematic reviews to generalize knowledge about MLOps practices necessary for the effective deployment of machine learning models.
2. Analyze the relationships between MLOps principles, processes, and practices.
3. Identify the most effective MLOps practices for deploying machine learning models.

2. Meta-synthesis of MLOps practices

2.1. Main concepts of the study

DevOps (Development & Operations) is becoming increasingly widespread, and companies are applying its methods in various fields [6]. In this context, *MLOps* (Machine Learning & Operations) automates ML workflows, such as pipelines, applying DevOps practices (implementing continuous integration/continuous deployment (CI/CD) for machine learning projects) [12, 6, 8].

According to Calefato et al. [12], key MLOps practices can be realized using GitHub Actions and CML (Continuous Machine Learning). While some workflows automate ML tasks with GitHub Actions and CML, production-grade, end-to-end MLOps pipelines appear rare in the analysed open-source GitHub projects. Practices focus more on reporting and metrics rather than retraining or deployment.

2.2. Research methodology

Systematic reviews can provide syntheses of the state of knowledge in a field, from which future research priorities can be identified; they can address questions that otherwise could not be answered by individual studies; they can identify problems in primary research that should be rectified in future studies; and they can generate or evaluate theories about how or why phenomena occur [13, p. 1]. The main aim of a systematic review is facilitating evidence-based decision-making [13, p. 6]. The main difference between systematic and literature reviews is the "place of idea". The literature review can be idea-driven; thus, all sources can be selected to confirm some idea. Indeed, the systematic review is the scientific method; instead of ideas, we operate with research questions and hypotheses. As a result, the systematic review can produce new evidence-based knowledge.

On 23 February 2024, we proceeded with a search request to the Scopus database by article title:

```
TITLE ( ( systematic OR review OR survey ) AND mlops )
```

5 documents were found (table 1), of which 3 [6, 8, 5] relate to systematic reviews.

Table 1: Results of searching for existing systematic reviews in Scopus.

Bibliographic description	Review content
G. Recupito, F. Pecorelli, G. Catolino, S. Moreschini, D. D. Nucci, F. Palomba, D. A. Tamburri, A Multivocal Literature Review of MLOps Tools and Features, in: 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2022, pp. 84–91. doi:10.1109/SEAA56994.2022.00021.	Recupito et al. [6] conducted a "multivocal" literature review – a kind of systematic review that uses both "white" (articles, book chapters, etc.) and "grey" sources (blog posts, technical documents, videos, etc.). The authors' aim was to identify tools for creating MLOps pipelines and analyze their main characteristics and features. The authors investigated the functionality of 13 MLOps tools and showed that most MLOps tools support the same features but apply different approaches that can provide different advantages depending on user requirements.

Continuation on the next page

Continuation of table 1

Bibliographic description	Review content
A. Lima, L. Monteiro, A. P. Furtado, MLOps: Practices, Maturity Models, Roles, Tools, and Challenges – A Systematic Literature Review, in: Proceedings of the 24th International Conference on Enterprise Information Systems - Volume 1: ICEIS, INSTICC, SciTePress, 2022, pp. 308–320. doi:10.5220/0010997300003179.	Lima et al. [8] conducted a systematic literature review to identify practices, standards, roles, maturity models, challenges, and tools of MLOps. 30 articles were selected for analysis. The study results allowed to conclude that MLOps is still at an initial stage.
C. Haertel, D. Staegemann, C. Daase, M. Pohl, A. Nahhas, K. Turowski, MLOps in Data Science Projects: A Review, in: 2023 IEEE International Conference on Big Data (BigData), 2023, pp. 2396–2404. doi:10.1109/BigData59044.2023.10386139.	Haertel et al. [14] provided an overview of MLOps applications in Data Science projects. The authors showed that when considering contemporary MLOps approaches, the emphasis is placed on model development and deployment, while organizational aspects (business understanding, evaluation) receive insufficient attention. Since Data Science project success does not exclusively depend on technical matters, the authors propose that future research should continue to advance the MLOps field by bridging the gap between business objectives of the organization and how these objectives are represented and modelled using appropriate concepts.
R. Cohen, Digital Strategy, Machine Learning, and Industry Survey of MLOps, in: Digital Strategies and Organizational Transformation, 2023, pp. 137–150. URL: https://tinyurl.com/33z6zpd3. doi:10.1142/9789811271984_0008. [15]	As part of a digital strategy, machine learning (ML) has become a common toolset and capability across many businesses. However, the operational aspects of machine learning (MLOps) are often overlooked for ML projects until they are already installed and being executed in the business environment. This chapter provides a review of MLOps products and vendors to give data scientists the ability to set up the appropriate ML infrastructure in a proactive manner.
J. Diaz-de Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, A. Almeida, A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey, ACM Comput. Surv. 56 (2023) 84. doi:10.1145/3625289.	Diaz-de Arcaya et al. [5] analyze the challenges, opportunities, and perspectives of implementing MLOps and AIOps. The authors analyzed the open issues, opportunities, and trends faced by organizations when implementing MLOps and AIOps, the frameworks and architectures, as well as the fields of their use. The systematic review of 93 studies provided an opportunity to identify: 1) successful implementation of artificial intelligence projects requires a collaborative culture and a combination of software engineering, data science, and DevOps skills; 2) containerization, data and model versioning, FaaS (Function-as-a-Service) and serverless architectures are useful for supporting the MLOps/AIOps lifecycle; 3) monitoring the environment is important for retraining and redeploying components; 4) AIOps is used predominantly in complex environments, such as 5G and 6G technologies, while MLOps is more common in traditional industrial environments.

The papers by Haertel et al. [14] and Cohen [15] are not systematic reviews, but the results obtained in these works were taken into account when performing the meta-synthesis [16] for combining (appendix B) and thematic analysis of the results obtained in the systematic reviews.

The meta-synthesis was performed according to Chrastina [17, pp. 123-125]:

1. *Defining the subject of research*: MLOps practices for the effective deployment of models.
2. *Identifying relevant sources*: systematic literature reviews [6, 8, 5] and a review of MLOps products and providers [15].
3. *Thorough study* to determine the common time period for the analyzed works, similarities and differences: in the aim, research questions, sources, inclusion, exclusion, and quality criteria, definitions of MLOps and MLOps stages.

4. *Defining the relationship between works* through identifying and grouping key topics.
5. *Mutual translation of results of different works* through defining common terminology, explaining contradictions in the results from different works, and generalizing results from different works.
6. *Synthesis of results.*
7. *Publication of the meta-synthesis.*

2.3. Thorough study and defining the relationship between works

2.3.1. Distribution of reviews by year

Two works [6, 8] refer to 2022, one [5] – to 2023. At the same time, the sources analyzed in [6] are limited to 2020, in [8] – to 2021, and in [5] – to 2023. In addition, the work [5] mentions the work [8] as a previous one.

2.3.2. Review objectives

The aim of the meta-synthesis of the review objectives [6, 8, 5] was to identify common and different aspects regarding the general focus and tasks of these studies.

Common aspects of the objectives of the considered reviews are:

1. All reviews [6, 8, 5] are aimed at researching and generalizing knowledge about the MLOps methodology, its practices, tools, and challenges.
2. The reviews [6, 8] aim to identify and analyze MLOps tools used to automate the development and deployment processes of machine learning models.
3. The reviews [5, 8] seek to provide an understanding of the general state of MLOps practices implementation in industry and academia.

Distinct aspects of the review objectives are:

1. The review [6] focuses more on identifying and analyzing the functional capabilities of MLOps tools for creating machine learning pipelines.
2. The review [8] pays attention to a wider range of MLOps aspects, such as practices, roles, maturity models, challenges, in addition to tools.
3. The review [5], in addition to the MLOps methodology, also considers the related concept of AIOps. More attention is paid to highlighting the opportunities, challenges, and future trends in both areas.

Thus, despite some differences in focus and breadth of coverage, all the considered reviews are united by a common goal – to investigate and generalize knowledge about the MLOps methodology, its practical application, tools, challenges, and state of implementation to promote further development of this area.

2.3.3. Review research questions

The aim of the meta-synthesis of the research questions of the reviews [6, 8, 5] was to identify common and different aspects regarding the main directions of research within the study of the MLOps methodology.

Common aspects of the research questions of the considered reviews are:

1. All reviews [6, 8, 5] contain questions about tools and platforms used to implement MLOps practices, automate development processes, deployment, and monitoring of machine learning models.

2. The reviews [8, 5] include questions regarding challenges and open problems faced by organizations when implementing MLOps.

3. The reviews [8, 5] consider questions about opportunities and future trends in the field of MLOps.

Distinct aspects of the research questions of the reviews are:

1. The review [6] contains more specific questions about the functional capabilities and features of MLOps tools for creating machine learning pipelines.

2. The review [8] includes questions regarding the roles and responsibilities of specialists involved in MLOps implementation, as well as maturity models for assessing the level of automation of model deployment processes.

3. The review [5], in addition to MLOps, also considers questions specific to the AIOps methodology and pays attention to current and future areas of application of these approaches.

Thus, the research questions of the considered reviews cover a wide range of MLOps aspects, from tools and platforms to challenges, opportunities, and areas of application. Despite some differences in the focus of the questions, all reviews seek to explore the key components and factors that influence the implementation and development of MLOps practices in organizations.

2.3.4. Review information sources

The aim of the meta-synthesis of the information sources of the reviews [6, 8, 5] was to identify common and different aspects regarding the databases, search engines, and types of literature used to search for relevant studies.

Common aspects of information sources of the considered reviews are:

1. All reviews [6, 8, 5] used electronic databases of scientific publications to search for relevant studies.

2. The reviews [6, 5] included both academic (peer-reviewed) and non-academic ("grey") literature sources such as blogs, websites, videos, code repositories, etc. in the search.

Distinct aspects of information sources of the reviews are:

1. The review [6] used Google Scholar to search for scientific publications and regular Google search for "grey" literature.

2. The review [8] limited the search to only academic databases such as ACM Digital Library, IEEE Xplore, ScienceDirect, and SpringerLink.

3. The review [5] used several databases (Scopus, arXiv, Springer, IEEE), but the main source was the Scopus database from Elsevier.

Thus, the considered reviews demonstrate different approaches to the selection of information sources. Some studies [6, 5] include both academic and non-academic sources to obtain a more complete picture of the practical application of MLOps. Others [8] focus exclusively on peer-reviewed scientific publications. The choice of sources can influence the coverage and type of studies found, and hence the results and conclusions of the reviews.

2.3.5. Criteria for including information sources in reviews

The aim of the meta-synthesis of the criteria for including information sources in the reviews [6, 8, 5] was to identify common and different aspects regarding the requirements that studies must meet for inclusion in the analysis.

Common aspects of the inclusion criteria in the considered reviews are:

1. All reviews [6, 8, 5] included studies that directly relate to the topic of MLOps, its practices, tools, and application.
2. The reviews [6, 8] considered studies that describe the experience, practices, architecture, or implementation of MLOps tools and processes.

Distinct aspects of the inclusion criteria in the reviews are:

1. The review [6] included studies that describe the components of the minimal MLOps lifecycle or present the experience and opinions of experts regarding MLOps.
2. The review [8] additionally included studies that assess the maturity of MLOps processes, consider roles and responsibilities in the ML model development lifecycle, and identify challenges in developing and implementing MLOps solutions.
3. The review [5] had more general inclusion criteria, considering studies published from 2018 to 2023 that contain new ideas and are closely related to the topic of MLOps and AIOps.

Thus, despite some differences, the inclusion criteria in the considered reviews mainly focus on studies that directly relate to MLOps, describe practical experience, tools, and processes, and consider various aspects of the ML model development lifecycle. The reviews [8, 5] have somewhat broader criteria, also including studies related to maturity assessment, roles, and MLOps challenges.

2.3.6. Criteria for excluding information sources from reviews

The aim of the meta-synthesis of the criteria for excluding information sources from the reviews [6, 8, 5] was to identify common and different aspects regarding the characteristics of studies that lead to their exclusion from the analysis.

Common aspects of the exclusion criteria in the considered reviews are:

1. The reviews [6, 8] excluded studies that do not provide sufficient details about the architecture, implementation, or application of MLOps tools and processes.
2. The reviews [8, 5] excluded studies published in languages other than English.

Distinct aspects of the exclusion criteria for information sources from the reviews are:

1. The review [6] excluded studies that promote commercial MLOps platforms without providing details on their implementation or use.
2. The review [8] excluded studies that relate only to the application of ML models without considering MLOps aspects, as well as short articles, posters, and studies without access to the full text.
3. The review [5] excluded studies with an insufficient number of citations (depending on the year of publication), as well as materials with limited access (by subscription) and articles published in insufficiently reliable sources.

Thus, the considered reviews apply different exclusion criteria to filter out studies that do not meet their requirements. The common aspect is the exclusion of studies with insufficient descriptions of MLOps processes and tools, as well as non-English publications. Differences lie in additional criteria, such as the exclusion of commercial platforms without technical details [6], short articles and posters [8], and materials with limited access and a low number of citations [5].

2.3.7. Quality criteria for information sources in reviews

The aim of the meta-synthesis of the quality criteria for the reviews [6, 8, 5] was to identify common and different aspects regarding the requirements for the quality and reliability of studies included in the analysis.

Common aspects of the quality criteria in the considered reviews are:

1. The reviews [8, 5] evaluated the quality of studies based on the completeness of the description of the methodology, context, and results.
2. The reviews [8, 5] considered the presence of substantiated evidence and arguments to support the study conclusions.

Distinct aspects of the quality criteria in the reviews are:

1. The review [6] used quantitative indicators of popularity (number of stars on Github, views on YouTube) to assess the quality and relevance of "grey" literature.
2. The review [8] evaluated whether the study presents empirical results, not just expert opinions, and whether the results are properly validated.
3. The review [5] used an extended set of criteria, including the presence of a comprehensive literature review, verification of results on use cases, the number of research questions addressed, the availability of open access, publication in high-impact journals, and the number of citations.

Thus, the considered reviews apply different approaches to assessing the quality of studies. The common aspect is the desire to include studies with a complete description of the methodology and substantiated results. However, the specific quality criteria differ: from the use of popularity indicators for "grey" literature [6], to the assessment of the empirical nature of the results [8] and consideration of bibliometric indicators such as journal impact factor and number of citations [5].

2.4. Mutual translation of results from different works and synthesis of results

For the mutual translation of the results from different works, in addition to systematic reviews [6, 8, 5], a review of MLOps products and providers [15] was involved.

2.4.1. Definition of MLOps

The aim of the meta-synthesis of MLOps definitions in the reviews [6, 8, 5, 15] was to identify common and different aspects in the understanding and interpretation of this concept.

Common aspects of MLOps definitions in the considered reviews are:

1. All reviews [6, 8, 5, 15] consider MLOps as a set of practices, principles, and processes for automating and managing the lifecycle of machine learning models.
2. The reviews [6, 5] emphasize the use of approaches and practices from DevOps in MLOps, such as continuous integration, delivery, and monitoring.
3. The reviews [8, 15] emphasize the role of MLOps in operationalizing machine learning solutions and transferring them to industrial operation.

Distinct aspects of MLOps definitions in the reviews are:

1. The review [6] focuses more on the technical aspects of MLOps, such as model lifecycle management, pipeline automation, and performance monitoring.
2. The review [8] considers MLOps as a set of practices specifically for operationalizing data science solutions.

3. The review [5] emphasizes the use of software engineering and machine learning principles in MLOps to create model-based products.

4. The review [15] considers MLOps as a separate area that focuses on automating the machine learning model lifecycle as part of companies' digital strategies.

Thus, despite some differences in emphasis and wording, all the considered reviews define **MLOps** as an **approach for managing, automating, and operationalizing the processes of developing, deploying, and supporting machine learning models based on practices from software engineering and DevOps**. MLOps is a key component for the successful implementation of machine learning solutions in an industrial environment.

2.4.2. Stages of the MLOps workflow

The aim of the meta-synthesis of the stages of the MLOps workflow in the reviews [6, 8, 5, 15] was to identify the most common steps in the lifecycle of developing and implementing machine learning models.

Common stages of the MLOps workflow in the considered reviews are:

1. All reviews [6, 8, 5, 15] include the stages of data collection and processing, model development and training, and model deployment in the working environment.

2. The reviews [6, 5, 15] highlight the stage of monitoring the performance and degradation of deployed models as an important part of the MLOps workflow.

3. The reviews [6, 15] include the stage of retraining models based on new data or on a schedule as part of the MLOps lifecycle.

Distinct aspects of the MLOps workflow stages in the reviews are:

1. The review [6] provides a detailed breakdown of the workflow stages, including the steps of data extraction, analysis, cleaning, and transformation, as well as model validation.

2. The review [8] focuses less on detailed stages and more on general MLOps functions, such as data collection, transformation, model training, and implementation.

3. The review [5] groups stages into broader categories, such as data management, distributed training, deployment, and monitoring.

4. The review [15] additionally highlights the stages of generating predictions and managing models and data as part of the MLOps workflow.

Thus, despite different levels of detail and grouping, the considered reviews demonstrate general consistency regarding the main stages of the MLOps workflow. These stages cover the entire lifecycle of machine learning models, from data collection and processing to deployment, monitoring, and retraining of models. Differences in the presentation of stages reflect different approaches to structuring and describing the MLOps workflow.

2.4.3. Frameworks and architectures that facilitate MLOps implementation

The aim of the meta-synthesis of frameworks and architectures that facilitate MLOps implementation in the reviews [6, 8, 5, 15] was to identify the most common and effective approaches and technologies in this area.

Common frameworks and architectures that facilitate MLOps implementation, according to the considered reviews, are:

1. The reviews [6, 8, 5] highlight open-source platforms and frameworks such as MLflow, Kubeflow, and TensorFlow Extended (TFX) as key components of the MLOps ecosystem.

2. The reviews [6, 5, 15] emphasize the importance of using cloud computing platforms and services such as AWS, Google Cloud, and Azure to deploy and scale MLOps solutions.

3. The reviews [8, 5] note that architectures based on containerization (e.g., using Docker) and container orchestration (e.g., using Kubernetes) are key to ensuring portability and scalability of MLOps solutions.

Distinct aspects of the considered MLOps frameworks and architectures in the reviews are:

1. The review [6] additionally highlights MLOps pipeline orchestration platforms such as Apache Airflow, Jenkins, and Polyaxon.

2. The review [8] additionally mentions tools such as Kubeflow, Polyaxon, Comet.ml, Kafka-ML, MLModelCI for managing pipelines and deploying models.

3. The review [5] considers a broader range of architectural approaches, including the use of edge computing, serverless computing, and event-driven architectures.

4. The review [15] focuses primarily on proprietary platforms and solutions from commercial providers such as Iguazio, Domino Data Lab, Comet, and Valohai.

Thus, there are many frameworks and architectural approaches that facilitate MLOps implementation, from open platforms and libraries to commercial solutions and cloud services. Key factors are support for automation, scalability, portability, and integration with existing systems and tools. The choice of appropriate frameworks and architectures depends on the specific requirements and constraints of the organization, as well as the level of maturity of its MLOps processes.

2.4.4. MLOps tools for creating machine learning pipelines and operationalizing models

The aim of the meta-synthesis of MLOps tools for creating machine learning pipelines and operationalizing models in the reviews [6, 8, 5, 15] was to identify the most popular and functional tools in this area.

Common MLOps tools mentioned in the considered reviews are:

1. The reviews [6, 8] highlight MLflow as a popular open-source platform for managing the lifecycle of machine learning models, experiments, and deployment.

2. The reviews [6, 15] mention cloud platforms from major providers such as AWS SageMaker, Google Cloud AI Platform, Azure Machine Learning, as tools for operationalizing models.

3. The reviews [8, 5] note that containerization tools such as Docker and orchestration tools such as Kubernetes are often used to deploy models.

Distinct aspects of the considered MLOps tools in the reviews are:

1. The review [6] provides a detailed list of tools for different stages of the MLOps pipeline, including orchestration platforms (Apache Airflow, Jenkins, Kubeflow, Polyaxon, Seldon Core, etc.) and deployment (TensorFlow Extended).

2. The review [8] additionally mentions tools such as Kubeflow, Polyaxon, Comet.ml, Kafka-ML, MLModelCI for managing pipelines and deploying models.

3. The review [5] focuses more on general categories of tools, such as experiment management systems, data and model versioning, and infrastructure automation.

Table 2
Popular MLOps platforms and products, and associated providers (based on [15, p. 141]).

Platform/product	Provider	URL
MLflow	MLflow	https://mlflow.org/
Google Cloud AI	Google	https://cloud.google.com/products/ai
Kaggle	Kaggle	https://www.kaggle.com/
SageMaker	Amazon	https://aws.amazon.com/sagemaker/
Cloud-Native Toolkit	IBM	https://develop.cloudnativetoolkit.dev/resources/workshop/ai/
Iguazio MLOps Platform	Iguazio	https://www.iguazio.com/
Azure Machine Learning	Microsoft	https://azure.microsoft.com/en-us/products/machine-learning
Huawei Cloud ModelArts	Huawei	https://www.huaweicloud.com/intl/en-us/product/modelarts.html
SparkCognition Generative AI Suite	SparkCognition	https://www.sparkcognition.com/products/sparkcognition-generative-ai-suite
Comet	Comet	https://www.comet.com/site/
Grid.AI	Grid.AI	https://www.grid.ai/
Modzy ModelOps Platform	Modzy	https://github.com/modzy
Valohai MLOps Platform	Valohai	https://valohai.com/
HPE Ezmeral ML Ops	Hewlett Packard Enterprise	https://www.hpe.com/us/en/software/ezmeral-ml-ops.html
Domino Enterprise MLOps Platform	Domino	https://domino.ai/

4. The review [15] details the functionality of popular commercial MLOps platforms such as Iguazio, Domino Data Lab, Comet, Valohai, etc. (table 2)

Thus, there is a wide range of MLOps tools for creating machine learning pipelines and operationalizing models, from open platforms such as MLflow, to commercial solutions from cloud providers and specialized companies. The choice of specific tools depends on the needs and scale of the organization, as well as compatibility with the existing technology stack.

2.4.5. Main features offered by MLOps tools

The aim of the meta-synthesis of the main features offered by MLOps tools in the reviews [6, 8, 5, 15] was to identify the key capabilities and components of these tools.

Common features of MLOps tools, highlighted in the considered reviews, are:

1. The reviews [6, 8, 5] note that MLOps tools usually provide capabilities for tracking experiments, versioning models and data.

2. The reviews [6, 8, 15] emphasize the importance of automation and orchestration features of MLOps workflows, such as model training and deployment pipelines.

3. The reviews [6, 5, 15] indicate the presence in MLOps tools of components for monitoring the performance and degradation of deployed models.

Distinct aspects of MLOps tool features, considered in the reviews, are:

1. The review [6] provides a detailed classification of features into three categories:
   a) *general features related to all stages of the MLOps pipeline*:
      - open source support;
      - scalability and elasticity;
      - extensibility;

- cloud-agnostic or cloud environment support;
- metadata management;
- continuous integration and delivery (CI/CD);
- user interfaces: graphical (GUI), command line (CLI), application programming interface (API);

b) *data management features*:

- real-time data streaming;
- data storage;
- data analysis, cleaning, and transformation;
- data monitoring;
- metadata management;
- providing data access via API;

c) *model management features*:

- support for various machine learning libraries and frameworks;
- experiment tracking and model versioning;
- model registry;
- automatic hyperparameter optimization;
- model testing (A/B testing);
- anomaly and model drift detection;
- model performance monitoring;
- model metadata management;
- model deployment via API.

2. The review [8] additionally highlights features such as automatic hyperparameter optimization of models and mobility support for deployment in different environments.

3. The review [5] notes the importance of integrating MLOps tools with existing systems and supporting collaborative work of teams.

4. The review [15] details the features of commercial MLOps platforms:
   - *model development*: environment for data analysis, feature development, training, and model experiments;
   - *operationalization of model training*: creating reproducible pipelines for model training and testing;
   - *continuous model training*: automatic support for the frequency of model retraining based on schedule, events, or ad-hoc requests;
   - *model deployment*: packaging, testing, and deploying trained models in the production environment;
   - *generating predictions*: providing predictions or classifications in real-time or batch processing mode;
   - *monitoring model performance*: tracking the efficiency and degradation of models, warning about the need for retraining;
   - *data and feature management*: support for storing, processing, and accessing data and generated features.

Thus, MLOps tools provide a wide range of features to support the lifecycle of machine learning models, with a focus on automation, experiment tracking, versioning, monitoring, and model deployment. Some tools offer more specialized features, such as hyperparameter optimization or data management. The choice of a tool with an appropriate set of features depends on the specific needs and goals of the organization in the field of MLOps.

2.4.6. Ways of deploying machine learning models in production environments

The aim of the meta-synthesis of ways of deploying machine learning models in production environments in the reviews [6, 8, 5, 15] was to identify the most common and critical practices in this area.

Common ways of deploying machine learning models in production environments, according to the considered reviews, are:

1. The reviews [6, 5, 15] note that models are often deployed using container technologies such as Docker, which provides model mobility and isolation.
2. The reviews [6, 8, 15] indicate the prevalence of deploying models in cloud environments using platforms and services from major providers such as AWS, Google Cloud, and Azure.
3. The reviews [6, 5] note that models are often deployed as web services using REST API or other protocols to provide access to predictions in real-time.

Distinct aspects of the considered ways of deploying models in the reviews are:

1. The review [6] additionally describes deploying models using orchestration platforms (Apache Airflow, Jenkins, Kubeflow, MLflow, Polyaxon, Seldon Core, Valohai) to provide automatic scaling and container management.
2. The review [8] notes that some MLOps tools, such as MLflow, Kubeflow, and Kafka-ML, have built-in capabilities to facilitate model deployment in different environments.
3. The review [5] considers deploying models not only in the cloud but also on edge devices using specialized frameworks such as TensorFlow Lite and Core ML.
4. The review [15] provides a detailed description of the main stages and features of the machine learning model deployment process using CI/CD pipelines and support for different environments using commercial MLOps platforms:
    a) *creating a CI/CD pipeline for models*: MLOps platforms such as SageMaker, Azure ML, and Databricks allow creating CI/CD pipelines to automate the model deployment process, which includes the stages of building, testing, and deploying models, as well as tracking artifacts and version management;
    b) *supporting different deployment environments*: MLOps platforms typically support multiple deployment environments, such as development, testing, and production environments; models can be deployed in different environments using appropriate configurations and access policies;
    c) *model deployment process*:
        - trained models are packaged in a standardized format (e.g., Docker container) along with necessary dependencies;
        - the model goes through testing and validation stages to ensure its correctness and compliance with requirements;
        - after successfully passing the tests, the model is deployed in the target environment (when deploying in the production environment, additional security and monitoring measures may be applied);

d) *automation and orchestration of deployment*: MLOps platforms use automation tools such as Jenkins or GitLab CI/CD to ensure continuous integration and deployment of models, which can be configured to automatically trigger on certain events, such as updating the model code or the appearance of new data;

e) *monitoring and management of deployed models*: MLOps platforms provide tools for monitoring the performance and metrics of deployed models in real-time: in case of problems or model degradation, the platform can automatically initiate the process of retraining or rolling back to the previous version of the model.

Thus, the most common ways to deploy machine learning models in production environments are the use of container technologies, cloud platforms and services, and the deployment of models as web services. The choice of a specific approach depends on the requirements for latency, scalability, and availability of models, as well as the existing infrastructure and ecosystem of tools in the organization.

2.4.7. Maturity models for assessing the level of automation in deploying machine learning models

Lima et al. [8] refer to several maturity models for assessing the level of improvement in the development process of machine learning solutions:

1. The maturity model proposed by Amershi et al. [18], mentioned simultaneously in [8] and [5]. This model, based on the Capability Maturity Model (CMM) and Six Sigma methodology, checks whether the activity: (1) has defined goals, (2) is consistently implemented, (3) is documented, (4) is automated, (5) is measured and tracked, and (6) is continuously improved.

2. According to Dhanorkar et al. [19], organizations can be classified into three levels of maturity for developing machine learning solutions: (1) data-oriented, (2) model-oriented, (3) pipeline-oriented.

3. Lwakatare et al. [20] describe five stages of improvement in development practices: (1) manual process led by data science, (2) standardized process of experimental-operational symmetry, (3) automated ML workflow process, (4) integrated software development and ML workflow processes, and (5) automated and fully integrated CD and ML workflow process.

4. Akkiraju et al. [21] proposed an adaptation of the CMM model with the definition of five levels of maturity for each assessed capability: (1) initial, (2) repeatable, (3) defined, (4) managed, and (5) optimizing.

All systematic reviews [6, 8, 5] indicate that the level of automation of MLOps processes is one of the key factors in assessing the maturity of an organization in this area. Despite the fact that the considered reviews do not provide an exhaustive description of MLOps maturity models, they emphasize the importance of assessing the level of automation of model development, testing, and deployment processes as a key factor in the maturity of an organization in this area. The adaptation of existing software development maturity models to the specifics of MLOps can be an effective approach to assessing and improving machine learning processes in an organization.

2.4.8. Roles and responsibilities identified in the activities of operationalizing machine learning models

The aim of the meta-synthesis of roles and responsibilities identified in the activities of operationalizing machine learning models in the reviews [6, 8, 5, 15] was to identify the key participants in the MLOps process and their functions.

Common roles and responsibilities identified in the considered reviews:

1. All reviews [6, 8, 5, 15] mention the involvement of data scientists / data science researchers who are responsible for developing, training, and experimenting with machine learning models.

2. The reviews [6, 8, 5] highlight the role of data engineers / data providers who are involved in extracting, processing, transforming, and ensuring the quality of data for model training.

3. The reviews [6, 5, 15] note the importance of DevOps engineers, ML/MLOps engineers, and software engineers in operationalizing models, automating deployment processes, creating pipelines, and managing environments.

4. The reviews [6, 5] emphasize the role of managers, leadership, and business stakeholders in defining model requirements for deployment, decision-making, and supporting the MLOps strategy.

5. The reviews [6, 5] emphasize the role of managers, leadership, and business stakeholders in defining model requirements, decision-making, and supporting the MLOps strategy.

Distinct aspects of roles and responsibilities, considered in the reviews:

1. The review [8] additionally highlights the roles:
    - *domain specialist* has deep knowledge of the subject area, plays an important role in obtaining data and validating results);
    - *computational scientist/engineer* has high technical skills to prepare the environment for the operation of machine learning models;
    - *ML scientist/engineer* is responsible for designing new machine learning models, has in-depth knowledge of statistics and ML algorithms;
    - *provenance specialist* manages the supply of data in the lifecycle of developing machine learning solutions, has knowledge of both the subject area and machine learning;
    - *manager* assesses models before their publication;
    - *application developer* develops applications in which the created models will operate;
    - *deployment lead* assesses aspects related to infrastructure components when deploying ML models to production.
2. The review [5] mentions the role of subject matter experts in labeling data in specific domains.

Thus, despite some differences in the detail of roles, the considered reviews recognize the need to involve specialists from different areas – software development, data engineering, machine learning, subject matter experts, and management, for the successful operationalization of machine learning models. Close collaboration and communication between these roles is critical for implementing MLOps practices in organizations (figure 1).

2.4.9. Challenges encountered in deploying machine learning models in production environments

The aim of the meta-synthesis of challenges encountered in deploying machine learning models in production environments in the reviews [6, 8, 5, 15] was to identify the most common and critical problems in this area. In [6] and [15], specific challenges are not explicitly listed, but they can be determined indirectly based on the discussion of MLOps and automation of machine learning pipelines in [6] and the description of the various stages of MLOps and the need for appropriate tools in [15].

Common challenges identified in the considered reviews:

1. The reviews [6, 8, 5, 15] note the complexity of managing the machine learning model lifecycle, including versioning, tracking, and reproducibility of models and data, as well as the problem of ensuring scalability and performance of models in real-world conditions with large amounts of data and requests.

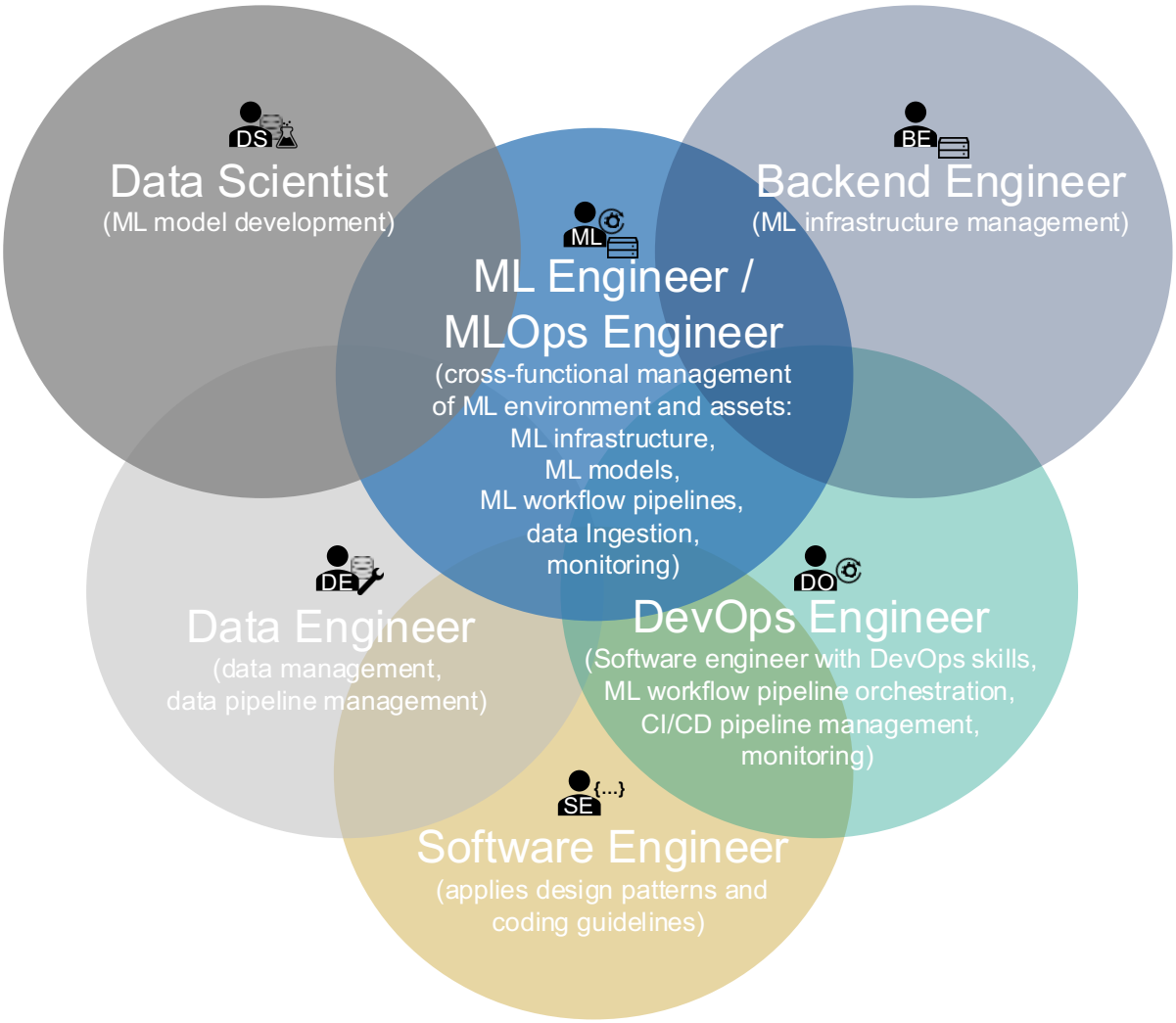

Figure 1: Intersection of roles and responsibilities (according to [2, p. 5]).

2. The reviews [8, 5, 15] point to challenges associated with monitoring and maintaining models in the production environment, including detecting data drift and model performance degradation.
3. The reviews [6, 8] consider challenges related to integrating software development with the machine learning pipeline.
4. The reviews [6, 5] consider challenges related to ensuring data security and privacy when deploying machine learning models.
5. The reviews [5, 15] indicate that the quality, availability, preparation, labeling, and integration of data from different sources is a significant challenge that requires a lot of time and resources, and highlight the problem of interpreting and explaining the results of model operation to end-users and business stakeholders.

Distinct challenges, considered in the reviews:

1. The review [6] emphasizes the need to automate all stages of the MLOps pipeline and integration with existing software development systems and processes.
2. The review [8] notes the problem of selecting and managing infrastructure for deploying models, including the choice between cloud and on-premises environments.

3. The review [5] notes the problems: a) the gap between software engineering and machine learning skills – data scientists often do not understand the requirements of certain production environments, and software developers do not have sufficient machine learning skills; b) effective distribution, parallelization, and orchestration of data and ML tasks; c) the diversity of computing infrastructure.

The considered reviews show that deploying machine learning models in production environments is associated with a number of challenges, such as managing the model lifecycle, ensuring scalability and performance, monitoring and maintaining models in real-world conditions. Addressing these challenges requires an integrated approach that includes automation of MLOps processes, selection of appropriate infrastructure, ensuring data security and privacy, and effective communication with business stakeholders.

2.4.10. Open issues, challenges, and peculiarities of MLOps

The aim of the meta-synthesis of open issues, challenges, and peculiarities of MLOps in the reviews [6, 8, 5, 15] was to identify the most relevant and promising areas of research and development in this field. In [6] and [15], open problems, challenges, and peculiarities of MLOps are not directly discussed, but they can be identified based on the analysis of MLOps tools and their capabilities in [6] and the description of the components and functions of MLOps platforms [15].

Common open issues and challenges of MLOps, identified in the considered reviews:

1. The reviews [6, 8, 5, 15] indicate the need to develop methods and tools to ensure the interpretability, reproducibility, and responsible use of machine learning models in the context of MLOps.

2. The reviews [6, 5, 15] emphasize the importance of developing approaches to data management in MLOps, including ensuring data quality, privacy, and security.

3. The reviews [6, 8] note the need to develop and implement MLOps standards and best practices to ensure consistency and compatibility between different tools and platforms.

4. The reviews [8, 5] emphasize the importance of the human factor in MLOps, including the need to ensure effective communication and collaboration between different roles and teams and the training of qualified personnel with cross-functional skills in programming, data processing, and operational activities.

Peculiarities of MLOps, identified in the considered reviews:

1. The review [6] notes that MLOps should take into account the specifics of the machine learning model development process, which differs from traditional software development.

2. The review [15] considers MLOps in the context of the overall digital strategy of the organization and emphasizes the need to align MLOps practices with business goals and needs.

Thus, the considered reviews identify a number of open issues and challenges in MLOps, such as the need to develop standards and best practices, ensure interpretability and responsible use of models, and effectively manage data. Peculiarities of MLOps, such as the difference from traditional software development, the importance of the human factor, and the need to integrate knowledge from different fields, require consideration when implementing MLOps practices in organizations.

2.4.11. Opportunities, future trends, and areas of application of MLOps

The aim of the meta-synthesis of opportunities, future trends, and areas of application of MLOps in the reviews [6, 8, 5] was to identify promising directions of development and potential areas where MLOps practices can bring significant benefits. In [6], they are not directly discussed, but they can be

determined indirectly based on the presented MLOps tools and their capabilities, it is possible to outline some potential directions and trends.

Opportunities and future trends of MLOps, identified in the considered reviews:

1. The reviews [6, 8, 5] note the potential for developing standardized MLOps platforms and tools that will simplify and accelerate the implementation of machine learning models in production.
2. The reviews [8, 5] note the prospects for integrating MLOps with other approaches, such as DataOps, ModelOps, and DevSecOps, to provide comprehensive management of the machine learning model lifecycle.
3. The review [8] points to: a) significant opportunities for further academic research and development due to the fact that MLOps is still at an early stage; b) the expectation of increasing demand for MLOps tools and platforms with the spread of artificial intelligence solutions; c) the emergence of new roles and competencies related to MLOps as the industry develops.
4. The review [5] points to: a) opportunities to apply MLOps practices in the context of distributed and federated model training, which will allow efficient use of decentralized data; b) involving business units and training leadership in MLOps principles; c) using hardware platforms such as FPGA and IoT to improve performance and privacy.

Current and future areas of MLOps application, identified in the considered reviews:

1. The reviews [6, 8, 5] note that MLOps is already actively used in industries such as finance, healthcare, commerce, marketing, and manufacturing, where machine learning models are used to solve real business problems.
2. The review [5] points to the potential of applying MLOps in the field of IoT and edge computing, where machine learning models can be deployed on resource-constrained devices, 5G and 6G technologies, educational and scientific activities.
3. The review [8] notes the prospects for using MLOps in transportation and logistics.

Thus, the considered reviews outline a number of opportunities and development trends for MLOps, such as creating standardized platforms, applying in the context of distributed learning, and integrating with other approaches to managing the lifecycle of data and models. Current and future areas of MLOps application include a wide range of industries, from finance and healthcare to IoT and natural language processing, which indicates a significant potential impact of this approach.

3. Analysis of MLOps practices

3.1. Relationship between MLOps principles, processes, and practices

MLOps is based on a set of principles [2, p. 3] and processes that ensure effective development, deployment, and support of machine learning models (MLOps practices).

MLOps principles define the fundamental foundations of designing machine learning pipelines:

- *automation*: maximum automation of all stages of the machine learning model lifecycle to reduce manual interventions and improve efficiency;
- *reproducibility*: ensuring the ability to reproduce the results of experiments and model deployment processes;
- *collaboration*: establishing effective collaboration and communication between different teams involved in model development and implementation;

- *continuous learning and improvement*: regular updating of models based on new data and feedback, continuous improvement of MLOps processes;
- *data governance*: ensuring data quality, security, and confidentiality throughout the model lifecycle.

MLOps processes define the sequence of actions for designing and implementing machine learning pipelines:

1. *Defining business goals and requirements*: aligning the goals of developing machine learning models with the business strategy of the organization.
2. *Data collection and preparation*: collecting, cleaning, transforming, and enriching data for model training.
3. *Model development and training*: selecting algorithms, developing model architecture, training and validating models.
4. *Model evaluation and testing*: evaluating model performance on test data, conducting tests for reliability, security, and compliance with requirements.
5. *Model deployment*: packaging models with necessary dependencies, deploying in target environments.
6. *Model monitoring and maintenance*: tracking model performance, identifying and resolving issues, updating models as needed.
7. *Model lifecycle management*: coordinating all stages of model development, deployment, and support, ensuring compliance with regulatory requirements.

MLOps practices define the most effective methods and technologies for implementing machine learning pipelines:

- *basic MLOps practices* include:
  - *continuous integration and delivery* (CI/CD): automation of the processes of building, testing, and deploying machine learning models;
  - *model and data versioning*: tracking changes in models and datasets, ensuring reproducibility of results;
  - *ML pipeline automation*: creating automated workflows for data collection, processing, model training, and evaluation;
  - *model performance monitoring*: tracking model quality metrics in the production environment, detecting performance degradation;
  - *experiment management*: organizing, tracking, and comparing different experiments with models and hyperparameters;
  - *model deployment*: packaging models with necessary dependencies, deploying in different environments (cloud, edge, etc.);
  - *model lifecycle management*: coordinating the processes of model development, testing, deployment, and monitoring to best ensure compliance with requirements;
- *additional MLOps practices* include:
  - *data security and privacy*: ensuring the protection of data used for model training, compliance with regulatory requirements;

- *model explainability and interpretability*: using methods and tools to understand and explain model behavior, especially in regulated industries;
- *data quality management*: monitoring and ensuring the quality of data used for model training and evaluation, detecting and handling anomalies;
- *configuration management*: versioning and managing configurations of environments where models are deployed, ensuring consistency across different environments;
- *model deployment strategies*: selecting and implementing appropriate deployment strategies;
- *infrastructure automation*: using Infrastructure as Code to automate provisioning and management of infrastructure for model training and deployment;
- *collaboration and communication*: establishing effective collaboration between data science, development, operations, and business units;
- *risk management and compliance*: identifying and mitigating risks associated with the use of machine learning models, ensuring compliance with regulatory requirements.

Figure 2 illustrates the relationships between key MLOps principles (light blue rectangles), main processes (green rectangles), and common practices (orange rectangles). Arrows show how principles influence processes, and processes, in turn, are implemented through specific practices. For example, the principle of automation influences all MLOps processes, from goal definition to model lifecycle management. The model development process is associated with practices such as versioning, pipeline automation, experiment management, and model interpretability.

3.2. CI/CD

CI/CD (Continuous Integration/Continuous Delivery) is a key element/practice/implementation of DevOps for automatic testing and deployment of code, data and models in a production environment (figure 3) [7, p. 7]. In MLOps, it is extended to automate the process of developing and deploying ML models, including the stages of building, testing, delivery, and deployment [2, pp. 3-4].

The CI/CD process in MLOps includes the stages of build, test, delivery, and deploy [2, p. 4]. However, unlike traditional CI/CD, MLOps may also have additional stages, such as model retraining.

CI/CD in MLOps is part of the MLOps system architecture and provides fast feedback to developers on the success or failure of certain stages, increasing overall productivity [2, pp. 3-4].

Typical triggers for starting the CI/CD process in MLOps on GitHub are git push and pull_request events [12, p. 4]. The events issue_comment, release, and schedule (on a schedule) can also be used. In the article Steidl et al. [7], potential main triggers were also investigated, such as feedback systems and alerts, scheduled orchestration service, traditional repository updates, and manual triggers. These triggers start the execution of the pipeline, which consists of four stages: (1) data processing, (2) model training, (3) software development, and (4) system commissioning. The data processing stage consists of a repetitive end-to-end lifecycle of data-related tasks, such as preprocessing, quality assurance, versioning, and documentation. The model training stage uses the results of data processing and illustrates model development tasks such as model design, training, quality assurance, collecting metadata for model improvement, version management, and documentation. After the pipeline completes model training, the software development stage prepares the model for deployment through packaging, quality assurance at the software level, and system versioning. In the final system commissioning stage, the model is deployed in a specific environment using different deployment strategies and the system is monitored [7, p. 21].

A feature of the CI/CD process in MLOps is the need to version not only code, but also data and models. This allows for reproducibility and the ability to roll back to previous versions [2, p. 4].

Among the popular tools for implementing CI/CD in MLOps are Jenkins [2, p. 3] and GitHub Actions [2, 12] and tools from cloud providers such as AWS CodePipeline, Azure DevOps Pipelines, etc.

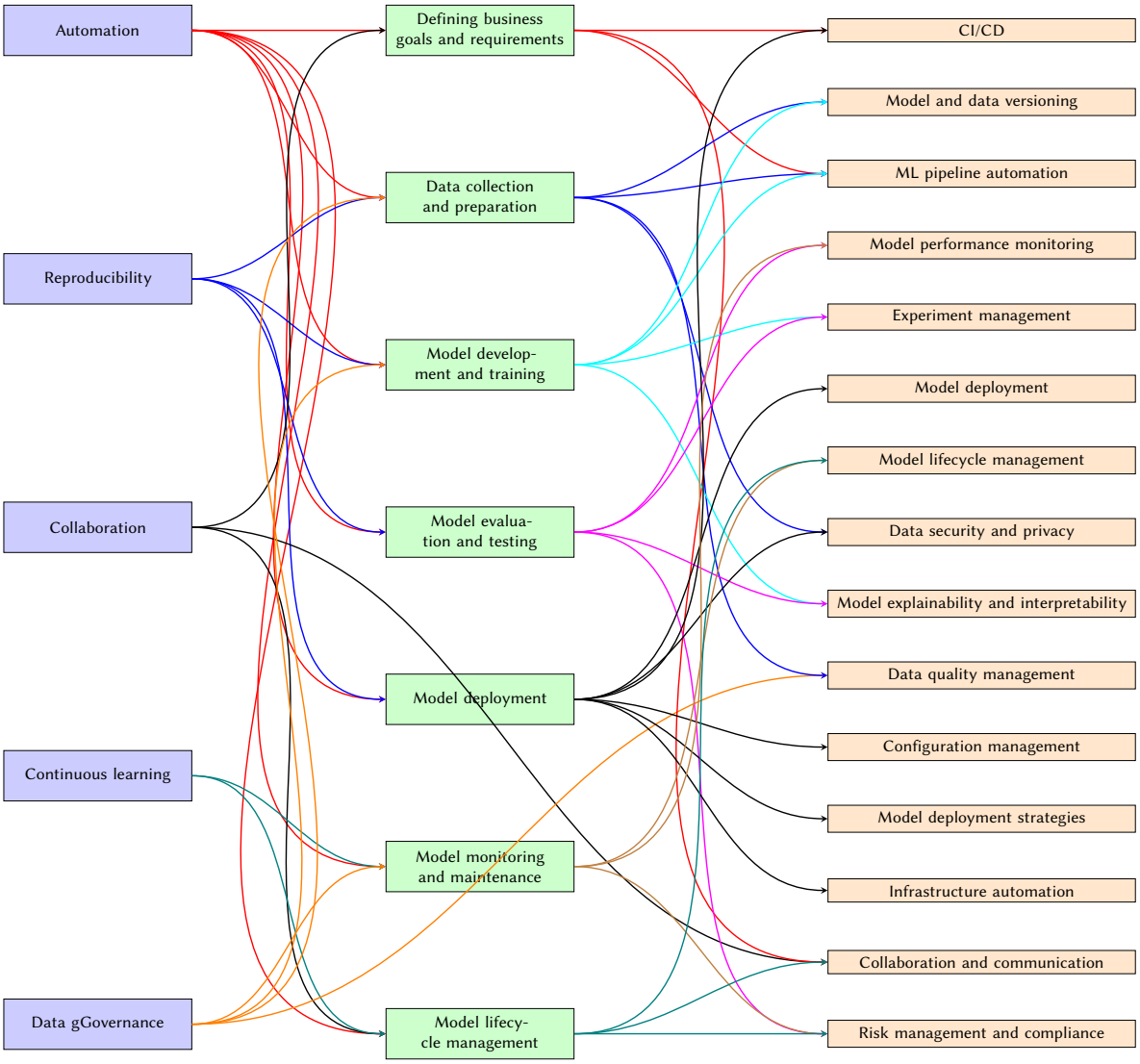

Figure 2: Diagram of relationships between MLOps principles, processes, and practices.

Figure 4 from the article by Kreuzberger et al. [2] presents an end-to-end MLOps architecture and workflow with functional components and roles involved at each stage. Let's consider in more detail each of the zones and stages depicted in the figure:

1. **A. MLOps Project Initiation**. At this stage, the business stakeholder (BS) analyzes the problem and defines the goal. The data scientist (DS) formulates the ML problem based on the business goal. The necessary data is also determined and an initial analysis is performed by the data engineer (DE) and data scientist (DS).
2. **Data Engineering Zone**. This zone includes two sub-stages:
   - **B1. Requirements for feature engineering pipeline**. Rules for transformation, cleaning, and calculation of new features are defined.
   - **B2. Feature Engineering Pipeline**. A data processing pipeline is implemented that receives data from various sources, applies transformations, and loads it into a feature store system.
3. **C. Experimentation**. At this stage, the data scientist (DS) performs data analysis, preparation, and validation, as well as model training and validation. The best model is saved in the Model Registry.

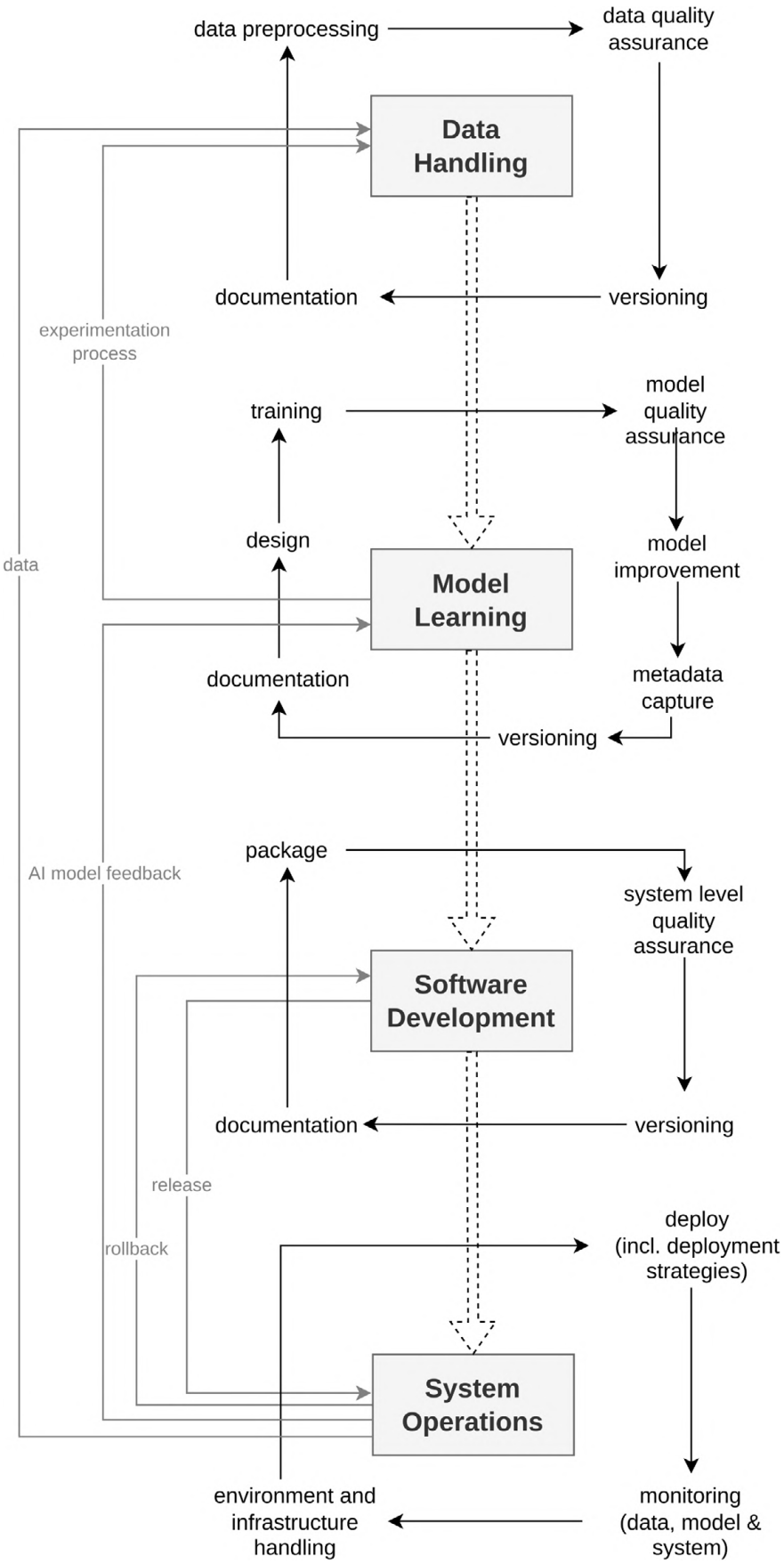

Figure 3: Continuous lifecycle pipeline for AI applications [7, p. 10].

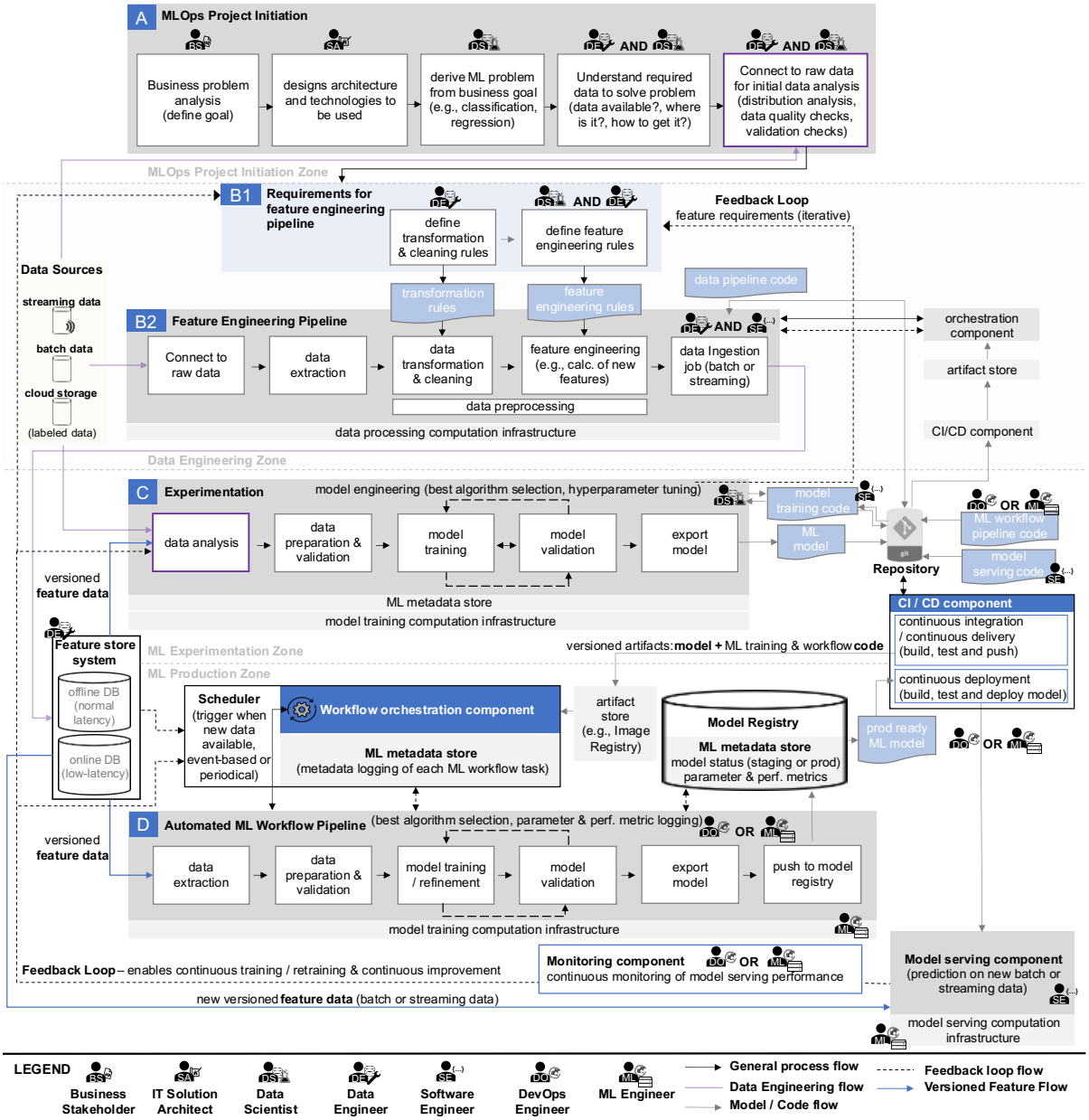

Figure 4: End-to-end MLOps architecture and workflow with functional components and roles [2, p. 6].

4. **ML Production Zone**. This zone includes an automated pipeline (D. Automated ML Workflow Pipeline), which ensures data preparation, training, validation, and registration of the model in production mode. The Model Serving component deploys the model, and the Monitoring Component ensures its continuous monitoring. In case of problems, information is transmitted through the Feedback Loop to initiate model retraining.

In addition to functional components, the figure also shows different roles and their areas of responsibility: Business Stakeholder (BS), Data Scientist (DS), Data Engineer (DE), DevOps Engineer (DO), ML Engineer (ML), Software Engineer (SE), and IT Solution Architect (SA).

3.3. Model and data versioning

Versioning is one of the key MLOps practices that ensures reproducibility and traceability of machine learning models [2, p. 3]. Versioning covers data, code, and the models themselves [7, p. 9]. Model

versioning not only captures model artifact versions but also model dependencies for tracking or reproducing different model versions that rapidly change over time [7, p. 13].

The purpose of data versioning is to guarantee the reproducibility of models and compliance with regulatory requirements. Data versioning can be implemented either by storing data snapshots or by referencing the original dataset. Since traditional version control systems cannot handle large amounts of data, specialized tools such as Data Version Control (DVC) are used [7, p. 11].

In the MLOps workflow, model versioning occurs at the model training stage. Its purpose is to save different versions of models along with their metadata for the ability to roll back to previous versions and reproduce results. The conditions for using model versioning are: (1) the constant evolution of models over time; (2) the need to track dependencies between models, data, and code [7, p. 12].

A feature of model versioning, as opposed to traditional code versioning, is the need to track a larger number of artifacts and metadata, as well as the need for larger amounts of memory due to the constant development of models [7, p. 13].

In addition to the data itself, versioning is subject to dependencies, data processing steps, and extracted features [7, p. 11]. For the latter, specialized feature stores are often used.

Model dependencies capture the relationship with related elements such as the corresponding dataset, source code, and configuration files. In addition, model versions store associated log files and model evaluation results. This allows checking whether model versions are constantly improving throughout the continuous lifecycle. Since versioning of artificial intelligence models is more complex and requires more memory due to continuous development, standard version control systems such as Git cannot be used as model repositories. Potential alternatives such as MLFlow, H2O, and DataRobot are container registries where image versions are stored, or model repositories that store model versions, including code, metadata, test results, and dependencies [7, p. 13].

3.4. ML pipeline automation

ML pipeline automation is a key MLOps practice that allows simplifying and accelerating the development, testing, and deployment of ML models in a working environment. This practice encompasses the automation of various stages of the ML pipeline, including data collection, preprocessing, model development, training, testing, validation, and deployment [7, p. 2].

The algorithm of an automated machine learning pipeline (figure 5) consists of the following stages:

1. Process initiation.
2. Extraction of versioned data from the storage.
3. Automated data preparation and validation.
4. Automated model training on new data (iteratively).
5. Model evaluation and hyperparameter tuning.
6. Model export.
7. Storing the model in the model registry.
8. Model deployment.
9. Model serving to obtain predictions.
10. Monitoring model performance.

The feedback loop ensures continuous improvement of the model by returning to the training stage. If necessary, monitoring can initiate a model retraining trigger, which starts the process from the beginning. This automated pipeline provides the ability to effectively manage the lifecycle of a

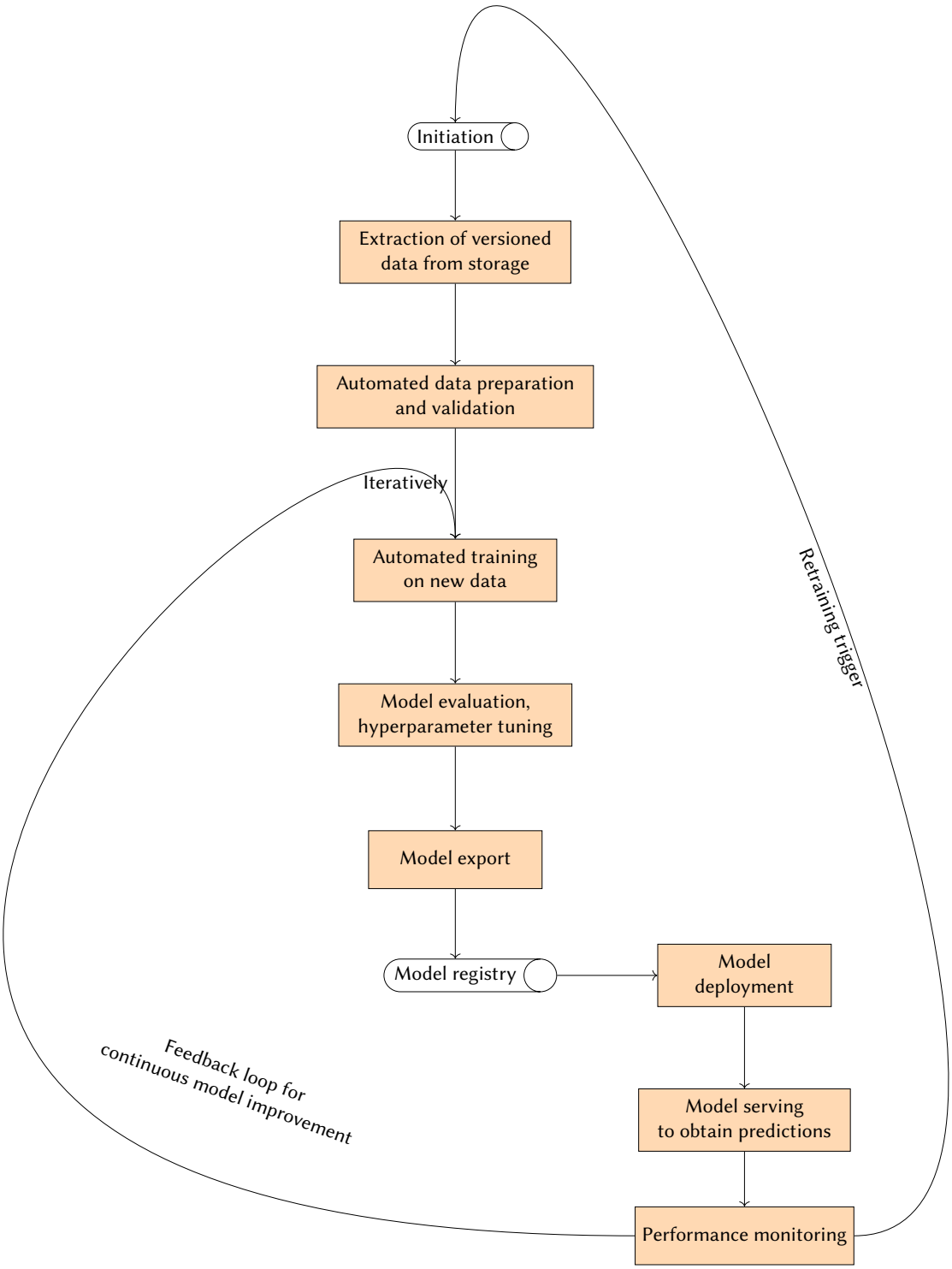

Figure 5: Algorithm of an automated machine learning pipeline (based on Kreuzberger et al. [2]).

machine learning model, from data preparation to deployment and monitoring, ensuring continuous improvement of the model. Features of using ML pipeline automation include the need to take into account the heterogeneity of models, frameworks, and execution environments. Therefore, each step of the ML pipeline should be as isolated as possible and have clear interfaces, for example, through the use of containerization [22, p. 6]. It is also critically important to ensure the ability to reproduce results and track artifacts [23, p. 1706].

Methods of ML pipeline automation include the use of pipeline management systems such as Kubeflow,

TFX [7, p. 19] or Apache Airflow [6, p. 4], as well as the development of custom automation scripts using tools such as Jenkins [24] or GitLab CI/CD [2, p. 2]. In this case, the pipeline is divided into separate steps, each of which is implemented as code or configuration, and then these steps are orchestrated and executed automatically [23, p. 1706].

3.5. Model performance monitoring

Monitoring the performance of machine learning models is an important MLOps practice that allows tracking the operation of models in a production environment, identifying problems, and taking measures to maintain their quality. This practice covers the collection of metrics regarding the operation of the model, real-time monitoring of these metrics, and alerts in case of detecting deviations from the norm [25, p. 128].

Model performance monitoring occurs at the model operation stage, after its deployment in the production environment. It is part of the continuous MLOps cycle and is performed in parallel with other practices, such as development, testing, and deployment [25, p. 127].

The conditions for using monitoring are the presence of a deployed ML model that processes real data and generates predictions. In addition, an infrastructure for collecting and storing monitoring data, for example, a database and visualization tools, should be set up [25, p. 134].

The monitoring process includes several steps (figure 6). First, key model performance metrics are defined, such as accuracy, error, latency, etc. Then, tools for collecting these metrics from the system where the model is deployed are set up. Next, the data is aggregated and visualized on dashboards for convenient analysis. Finally, alerts are set up that are triggered when metrics go outside of acceptable limits [25, p. 129].

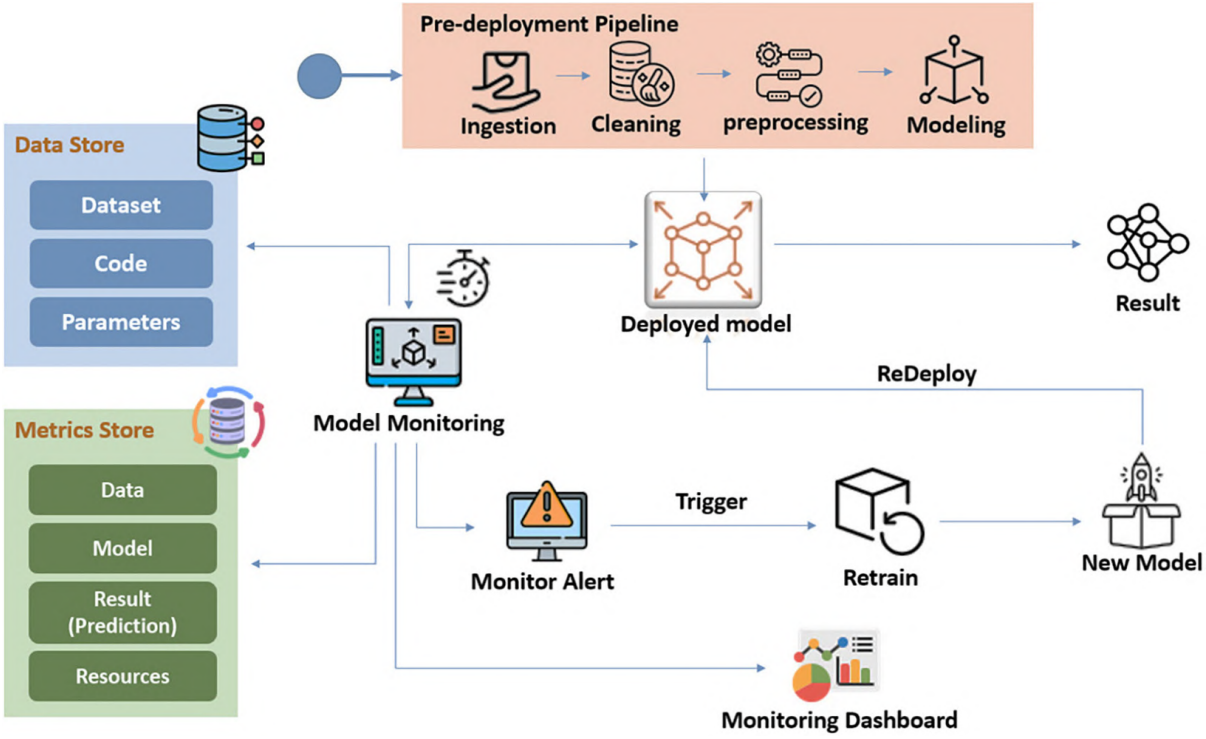

Figure 6: Monitoring process in MLOps [25, p. 129].

Figure 7 depicts the analogy proposed by Bodor et al. [25] between monitoring a machine learning system and an iceberg. This analogy emphasizes that the health of an ML system depends not only on visible elements, such as provided services, but also on hidden features that are difficult to track, such as data and the model itself:

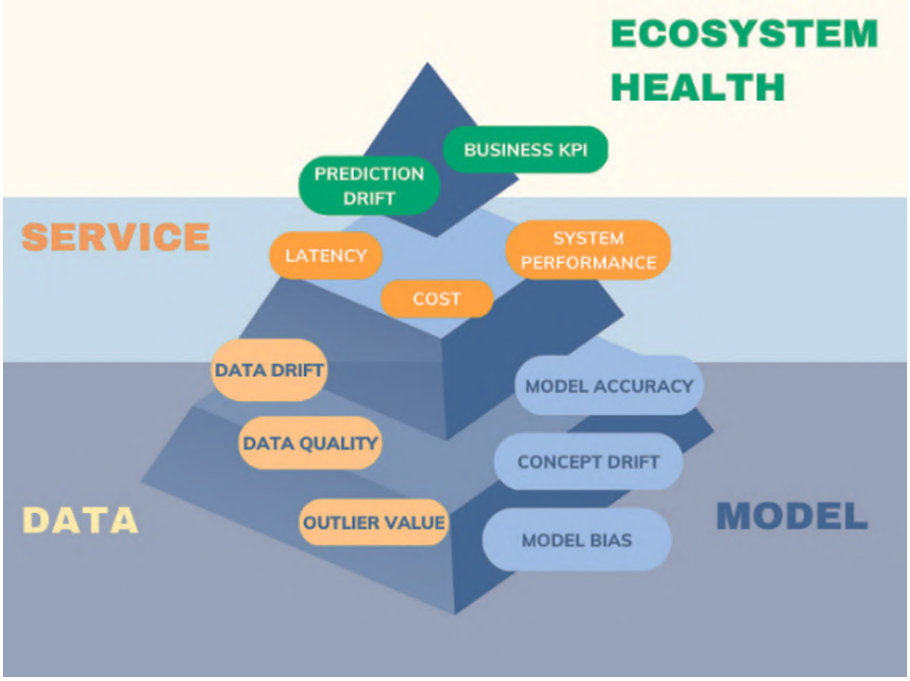

Figure 7: Hidden and visible characteristics associated with monitoring [25, p. 130].

- The top level of the iceberg represents ecosystem health, which includes aspects such as prediction drift and business key performance indicators (KPIs). The goal is to assess the performance, reliability, and stability of this top level.
- The service level includes metrics such as latency, cost, and overall system performance.
- The data level contains data quality characteristics, outlier values, and data drift.
- The lowest level of the iceberg is the model, which is characterized by accuracy, concept drift, and model bias.

A feature of monitoring in the context of MLOps is the need to track not only traditional software performance metrics (for example, CPU and memory usage), but also metrics specific to ML, such as model accuracy on new data (figure 8). In addition, MLOps involves automating the monitoring process and integrating it into the overall model development and deployment pipeline [25, p. 128].

There are a number of tools for setting up model performance monitoring in ML. They include open-source platforms such as Prometheus, Grafana, ELK stack, as well as commercial solutions from cloud providers, for example, AWS CloudWatch, Google Stackdriver, Azure Monitor [25, p. 135].

3.6. Experiment management

According to Singh [26], experiment tracking involves systematically storing metadata of machine learning experiments [26, p. 153].

Czakon and Kluge [27] defines that experiment metadata can include arbitrary scripts to run the experiment, environment configuration files, information about training and evaluation data, model parameter and training configurations, ML evaluation metrics, model weights, performance visualizations (e.g., confusion matrix or ROC curve), etc.

Experiment management (also known as experiment tracking) is part of MLOps focused on supporting iterative model development – the part of the ML project lifecycle where, in particular, hyperparameter tuning is performed to achieve the required level of model performance. Experiment management is closely intertwined with other aspects of MLOps, such as data and model versioning.

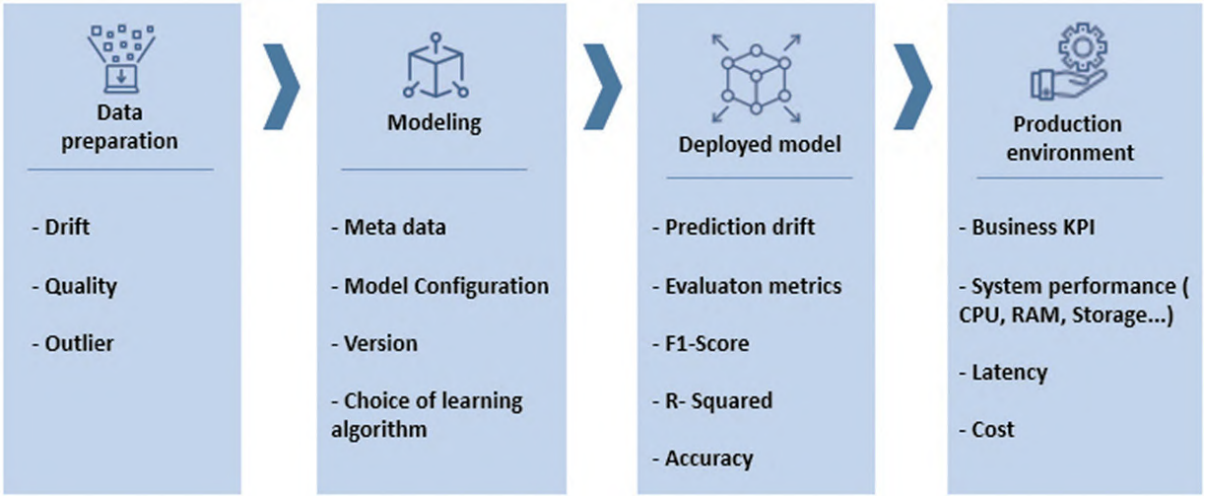

Figure 8: Elements for monitoring [25, p. 131].

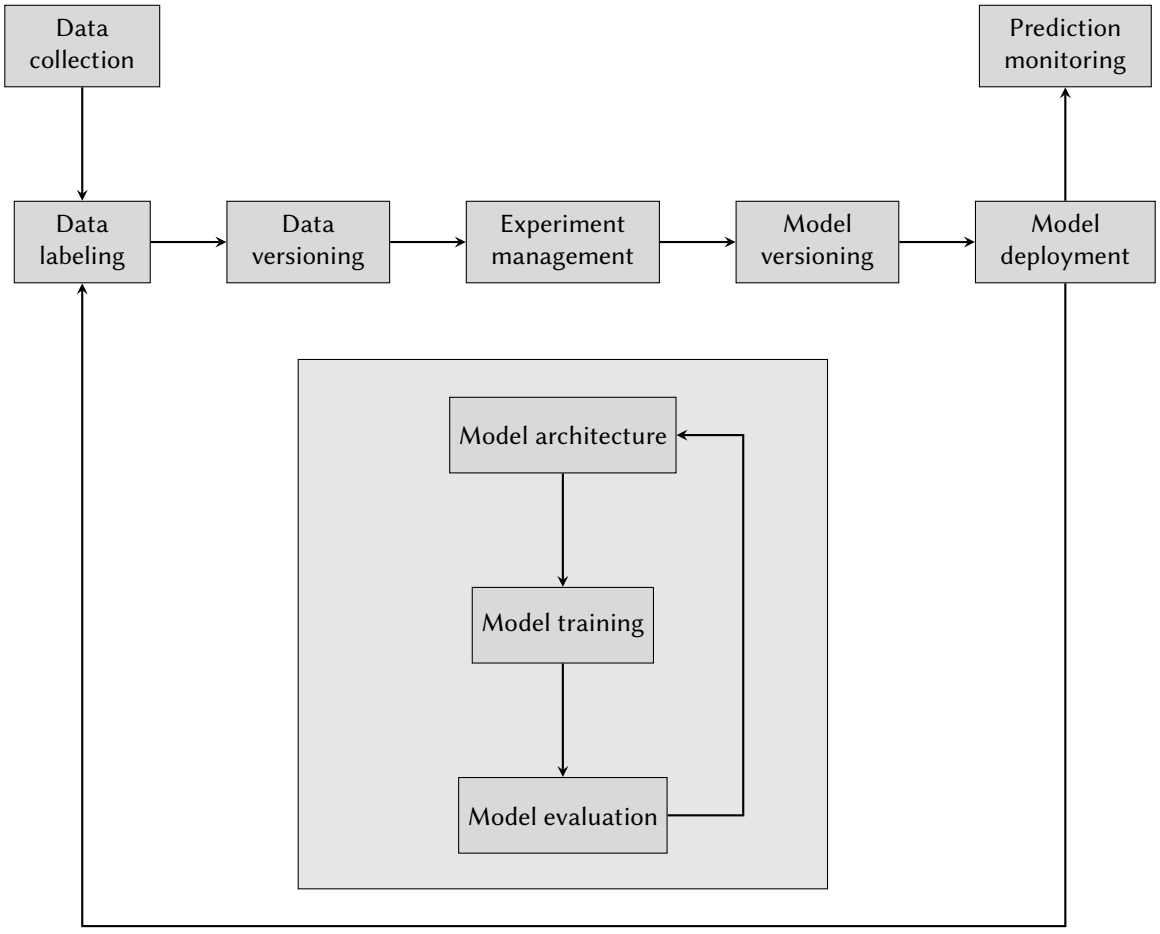

Figure 9: Experiment management in the MLOps lifecycle (adapted from Czakon and Kluge [27]).

The main condition for applying experiment management is the iterative nature of the model development and training process, when many experiments are conducted with different sets of hyperparameters, model architectures and training data (figure 9) [27]. This is typical for research and applied machine learning projects.

The most popular tools for experiment management are MLflow, Neptune.ai, Weights & Biases (wandb), Guild.ai, Comet.ml and TensorBoard [2]. They allow storing information about each experiment

in a repository – data used and its version, model parameters and architecture, performance metrics, model artifacts, etc.

3.7. Model deployment

Model deployment is an important MLOps practice that occurs at the operationalization stage in the workflow of developing and deploying ML models. This practice involves directly placing a trained and tested machine learning model into a production environment where it can be used to obtain predictions on real data [11, p. 1].

According to Kolltveit and Li [11], deployment, i.e. the transition of a packaged and integrated model into a service state, can occur in several different ways. Models packaged in containers are simply run directly as standalone services. However, models can be deployed in a target environment that is different from where they were packaged, and in this case, model transfer must occur using a push or pull pattern [11, p. 4]:

- in a pull-pattern deployment, the target environment (host application running, e.g., on a server or edge device) periodically polls for model updates and downloads them when available;
- in a push-pattern deployment, the target environment is notified of the availability of a new model by the master server (e.g., the server where the model was trained) through a messaging service, where the message contains metadata including the location of the updated model, or by initiating model transfer to the target environment through a specific receiving interface.

ML model deployment often occurs by packaging them in containers, for example, using Docker. This allows standardizing the deployment process and ensuring that the model will run in the same environment as during development and testing [11, p. 5].

There are different ways to deploy ML models depending on the requirements and architecture of the system [28, p. 68]:

- the model can be integrated directly into the application code;
- the model can be deployed as a separate service (microservice) with a REST API;
- the model can be loaded into a specialized environment for deploying and scaling ML models (for example, AWS SageMaker).

ML model deployment is associated with a number of problems, including ensuring low latency and high throughput for the prediction service [11, p. 5]. Various methods are used to solve them, such as adaptive queues with timeout for batch prediction, caching, dynamic switching between models of different accuracy, etc.

In terms of tools, container orchestration systems (Kubernetes), cloud provider services (AWS SageMaker, Azure ML), end-to-end MLOps platforms (MLflow, Kubeflow) are used for ML model deployment [28, 3].

ML model deployment is a critically important MLOps practice that allows transitioning developed models into a working state and using them for prediction. It occurs at the operationalization stage and requires considering a number of factors - from the way the model is packaged to ensuring the necessary performance of the prediction service. Deployment relies on modern tools for containerization, orchestration and infrastructure automation.

Based on the generalization [11, 28], a general scheme of machine learning model deployment was constructed (figure 10):

1. *ML Model* – a trained and tested machine learning model.
2. *Packaging* – the model is packaged in an appropriate format (e.g., a Docker container).
3. *Model registry* – the packaged model is placed in a model registry.

4. *Deployment* – the packaged model is deployed in the target environment (cloud, edge devices).
5. *Serving* – the model serves requests and generates predictions.
6. *Monitoring* – the performance of the model and environment is monitored. If necessary, model retraining is initiated.

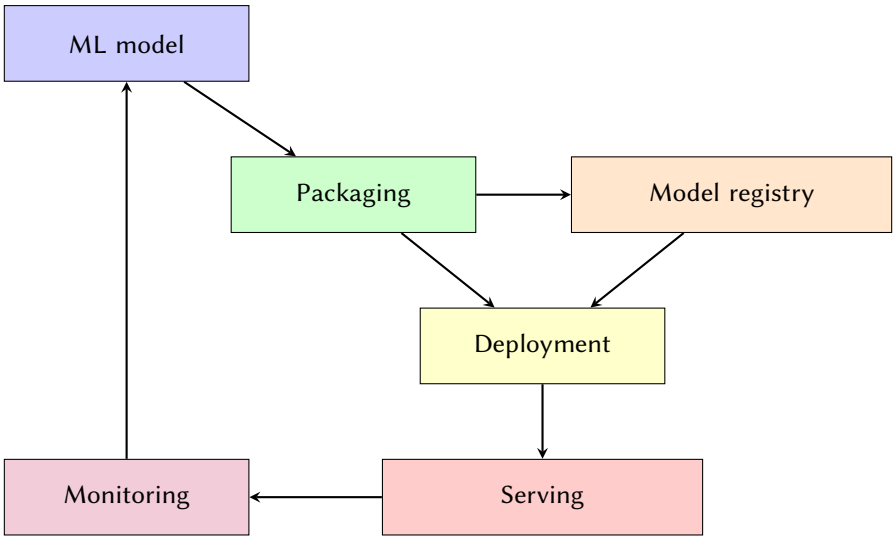

Figure 10: ML model deployment scheme.

3.8. Lifecycle management

According to Steidl et al. [7], continuous lifecycle management ((end-to-end) lifecycle management) is a continuous pipeline/flow management, which describes the (automatic) execution of specific tasks to ensure the management of the lifecycle of artificial intelligence [7, p. 7], which starts with data collection and ends with the deployment and monitoring of the artificial intelligence model [7, p. 2].

The goal of lifecycle management is to unify and standardize processes, which contributes to increasing the productivity of ML model development and their reliability in the industrial environment.

Key features of lifecycle management in MLOps:

- covers all stages: data collection and preparation, model development, training, validation and deployment [25, p. 127] (figure 11);
- is applied both at the early stages of model development and for their continuous support after deployment in the production environment [5, p. 9];
- involves versioning of data, code and models themselves to track changes and ensure reproducibility;
- involves monitoring model performance;
- automates processes using pipelines [7, p. 8].

Figure 11 shows the ML project development process (pipeline), which consists of a number of steps that are both linear and iterative (MLOps Pipeline block). The pipeline starts with data extraction, validation and preparation, then model training, evaluation and validation (experiment management). After that, the model is deployed in the production environment [25, p. 128].

An important aspect of the lifecycle is versioning of models and data, which occurs both in the experiment management pipeline and the production pipeline: the latter includes all stages of model

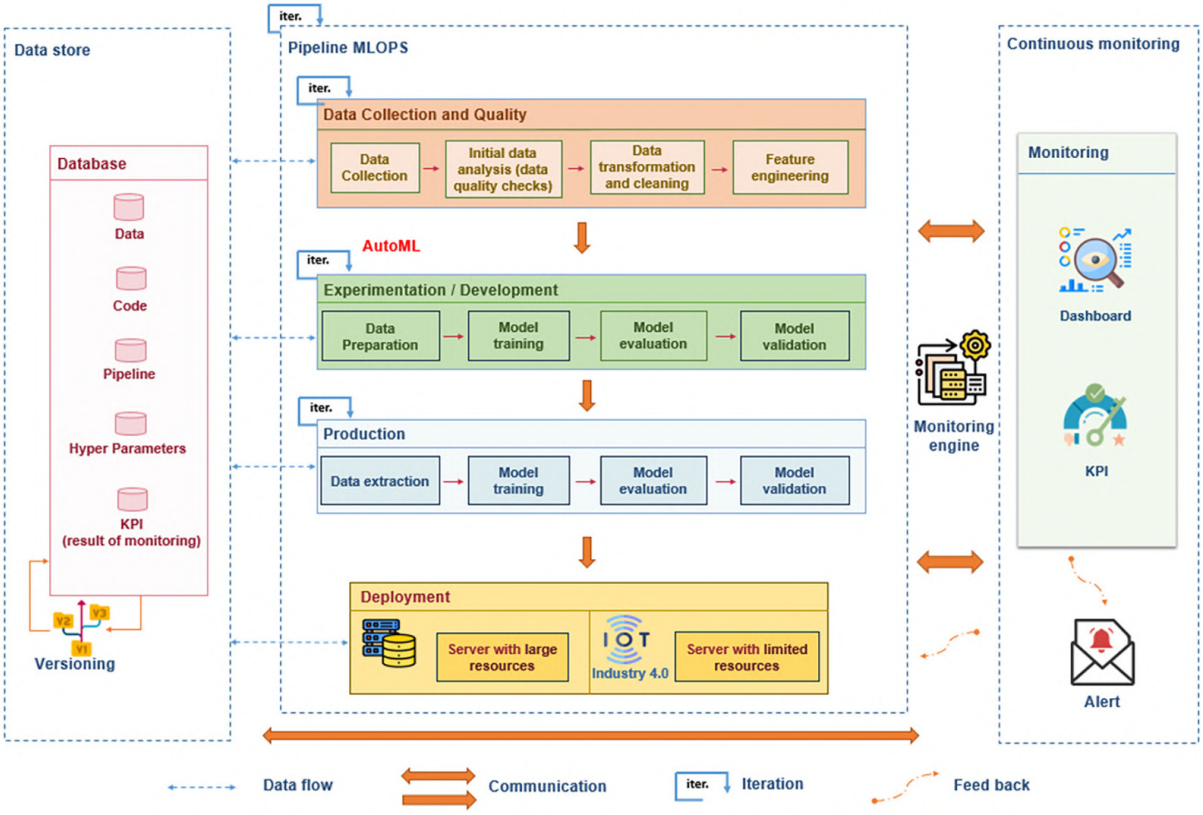

Figure 11: Machine learning project lifecycle in MLOps [25, p. 127].

creation, not just the final result of the experiment management pipeline. The iterative nature of MLOps provides the opportunity to obtain and maintain the best model, and the combination of monitoring based on key performance indicators and alert functions enables proactive intervention, ensuring the quality and reliability of deployed models throughout the lifecycle [25, p. 128].

The main tools used for lifecycle management in MLOps [25, p. 126]:

- platforms for organizing ML workloads and pipelines, such as MLflow, Kubeflow;
- data versioning systems, for example DVC (Data Version Control);
- tools for monitoring model performance in the production environment, such as Neptune.ai.

A comprehensive approach to lifecycle management in MLOps is implemented in Ease.ML (https://ease.ml/) – a lifecycle management system designed to simplify the entire development process [29]. The main goal of Ease.ML is to provide systematic recommendations and automation at all stages of the ML lifecycle, minimizing user effort.

Key features of Ease.ML:

1. *Human-in-the-loop process* – Ease.ML incorporates user interaction in a structured way, providing the ability for users to input data and make decisions at critical stages.
2. *Probabilistic data model* – the system uses a probabilistic database that handles uncertainty in data, which can arise from incorrect data, weak supervision, or other sources.
3. *Interactive environment* – Ease.ML uses Jupyter notebooks, allowing users to perform data manipulations, run ML operations, and visualize results in an integrated environment.
4. *Lineage graphs* – user interactions and operations are tracked in lineage graphs, which represent the entire ML workflow and ensure reproducibility.

5. *Automatic quality tuning and recommendations* – the system provides recommendations for model improvements based on errors detected in production, guiding users through data cleaning and collection tasks efficiently.

The Ease.ML process is divided into three sub-processes and consists of eight stages that cover the entire ML lifecycle:

- Day 0: Pre-ML Subprocess
  1. *Problem formulation* – clearly define the problem and ML goals.
  2. *Feasibility study* – assess the feasibility of an ML solution with the available data and resources.
- Day 1: AutoML Subprocess
  3. *Data preparation* – clean, preprocess and augment data to make it suitable for training ML models.
  4. *Model training*, using prepared data, with the use of AutoML to automate model selection and tuning.
  5. *Evaluation of trained model performance* on validation datasets to determine compliance with criteria.
  6. *Selection of the best model* and its deployment in the production environment.
- Day 2: Post-ML Subprocess
  7. *Continuous integration and delivery* – integration of the model into the production environment and setup of CI/CD pipelines to manage model updates and performance monitoring.
  8. *Model maintenance* through continuous monitoring of the model in the production environment and necessary updates or retraining of the model to adapt to new data or changing conditions.

3.9. Data security and privacy

Data security and privacy is the practice of protecting information and data used in machine learning processes from unauthorized access, misuse, and leakage [30, p. 1]. It aims to ensure the integrity, availability, and confidentiality of data at all stages of the MLOps lifecycle.

Data security and privacy should be considered at all stages of MLOps, from problem definition to monitoring of the deployed system. Ensuring security at the stages of data management, model development and deployment is especially critical, as this is where data is most vulnerable [31, p. 8].

The practice of data security and privacy is mandatory in cases where the ML system operates with sensitive data (personal, financial, medical, etc.) or is deployed in mission-critical environments (healthcare, automotive industry, manufacturing). However, even for less sensitive applications, an appropriate level of security must be ensured in accordance with regulatory requirements and user expectations.

Unlike traditional software development, in the context of MLOps, new attack vectors and vulnerabilities specific to machine learning emerge (figure 12). In particular, ML models are vulnerable to attacks such as data poisoning, model inversion, and adversarial example attacks [30, pp. 8-15]. This requires the use of specialized defense strategies.

To ensure data security and privacy in MLOps, both standard security tools (encryption, authentication, logging, etc.) and specialized ML-oriented solutions are used. Examples of the latter include libraries for secure federated learning aggregation, tools for testing models for vulnerabilities, frameworks for privacy-preserving machine learning based on homomorphic encryption or secure enclaves [31, pp. 10-11].

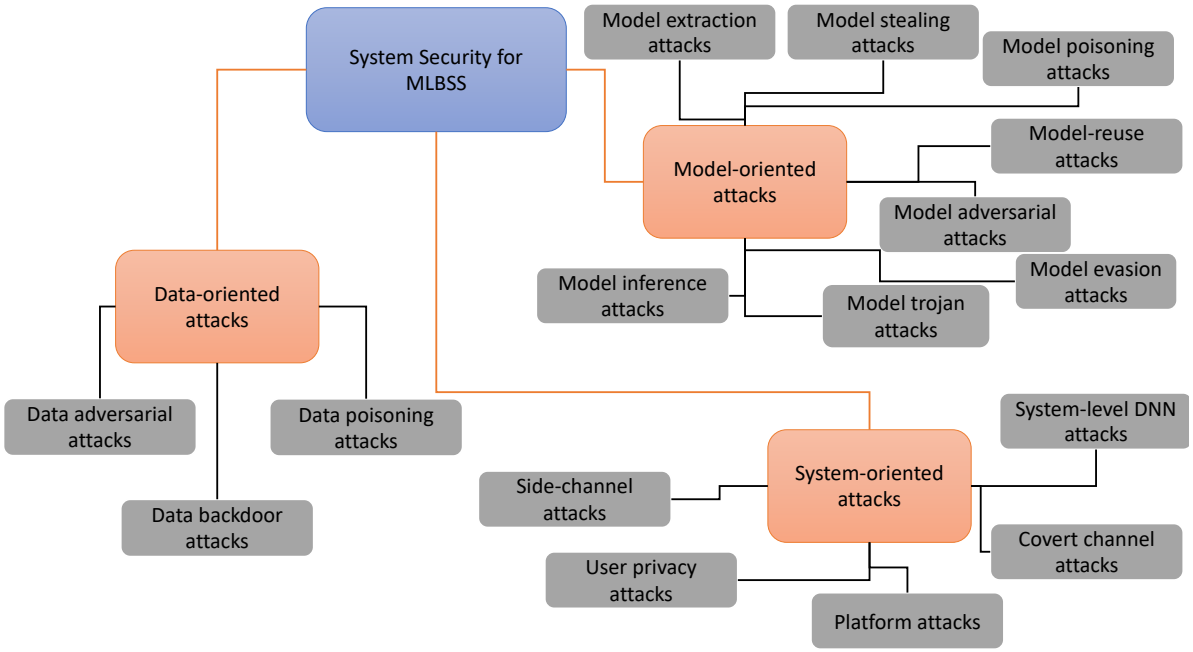

Figure 12: Classification of attacks on MLOps systems [30, p. 9].

Data security and privacy is ensured by implementing various control and protection mechanisms at each stage of MLOps. This includes, in particular, data encryption and anonymization, authentication and access control, integrity checking, activity monitoring, incident response, as well as regular security audits and penetration testing [30, pp. 4-5].

Figure 13 summarizes the main security components and practices in MLOps presented in [30, pp. 22-25].

Essential components of MLOps:

- problem definition includes understanding the problem and system requirements;
- data management covers data collection, labeling and verification;
- model construction and deployment includes model selection, building, optimization and evaluation, as well as its deployment in the target environment;
- system maintenance involves monitoring the functioning of the deployed system.

Security practices in MLOps:

1. At the problem definition stage:
   - risk assessment;
   - threat modeling.
2. At the data management stage:
   - data flow diagram;
   - STRIDE methodology for threat classification;
   - conceptual modeling;
   - data validation;
   - data linter;
   - data verification.

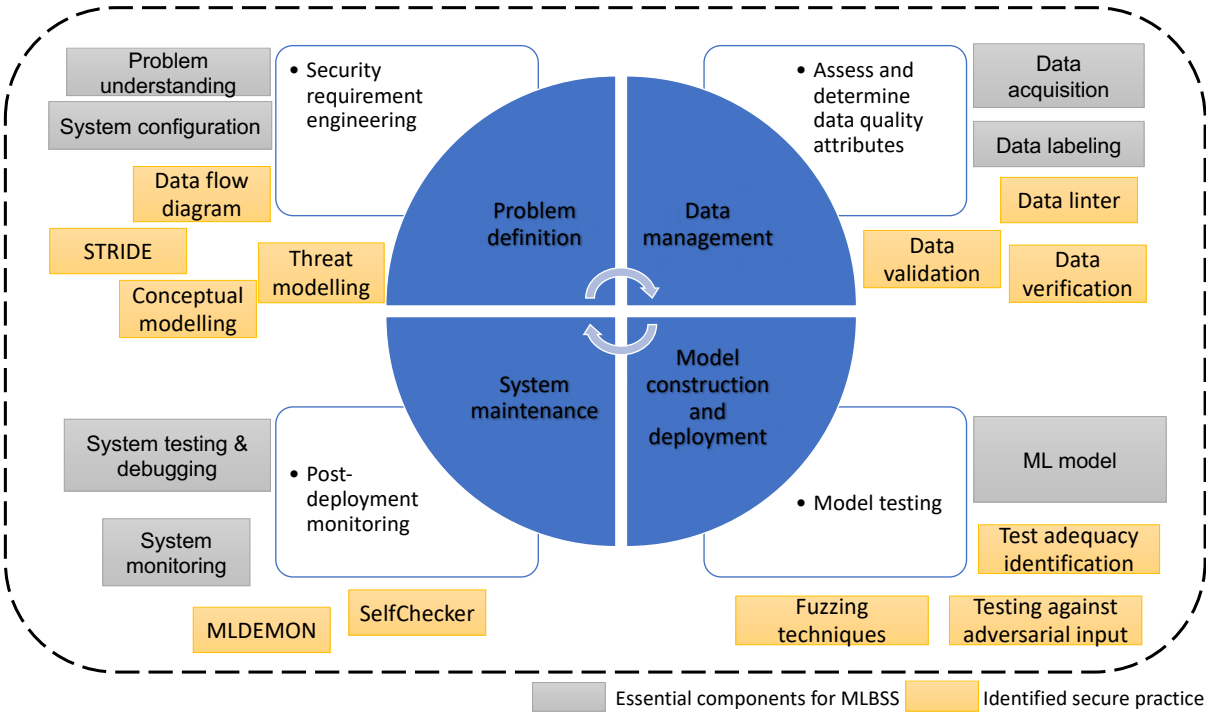

Figure 13: Main security components and practices in MLOps [30, p. 20].

3. At the model construction and deployment stage:
   - identification of test adequacy criteria;
   - testing against adversarial input;
   - fuzzing techniques.
4. At the system maintenance stage:
   - ML model monitoring tools (MLDEMON, SelfChecker);
   - static analysis;
   - exploratory attacks;
   - evasion attacks;
   - data poisoning attacks;
   - manual testing;
   - dynamic analysis.

Thus, security practices in MLOps are integrated into all main stages of the development lifecycle and include both general methods of security assessment and testing (threat modeling, static/dynamic analysis) and specialized approaches focused on the peculiarities of machine learning systems (testing against adversarial examples, detection of data poisoning).

3.10. Model explainability and interpretability

Explainability/interpretability of models is an important MLOps practice for ensuring transparency and trust in machine learning models. Explainability refers to the ability to explain and understand the decision-making processes of a machine learning model [10, p. 66]. This is especially important when the model is used to make decisions that have significant consequences, such as in military operations or law enforcement activities [9].

Explainability as the basis for trust in the ML project allows users to trust the prediction, which increases transparency. The user can verify which factors contributed to certain predictions, introducing an additional layer of accountability. The terms “explainability” and “interpretability” are often used interchangeably, but for MLOPs, explainability is more than interpretability in terms of importance, completeness, and fidelity of predictions or classifications (figure 14) [4, p. 63608].

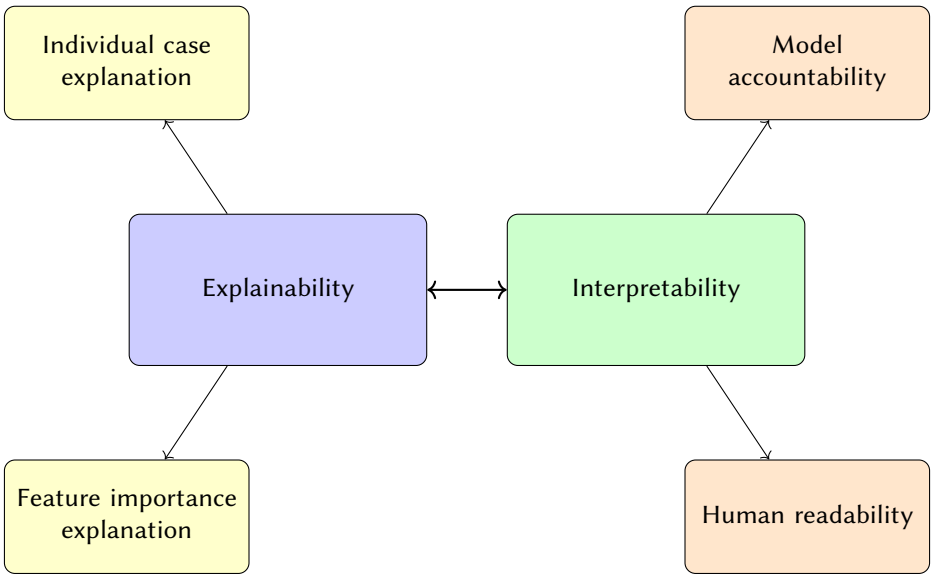

Figure 14: Relationship between explainability and interpretability in MLOps.

Explainable Artificial Intelligence (XAI) is a research direction that promotes explainable decision making [4, p. 63609]. Explainability can be defined as the degree to which an observer can understand the cause of a decision [32, p. 8]. An ML system is explainable when it is easier to identify causal relationships between the inputs and outputs of the system. The more explainable a model is, the better practitioners understand the internal business procedures that occur during model decision making. An explainable model does not necessarily translate into a model that a human can understand (internal logic or processes underlying it), but the explainability of the model allows the user to strengthen trust in the predictions made by the deployed system [4, pp. 63614-63615].

Various methods and tools can be used to achieve model explainability, such as:

- attribution-based methods (integrated gradients, saliency maps) or perturbation-based methods (SHAP) [33, p. 3], which explain the model’s decision by assigning high scores to the most influential input features;
- incorporating an attention mechanism into models, which allows focusing on the most relevant network states and inputs [33, p. 2];
- using a combined reward signal during training, which includes not only target metrics but also interpretability metrics [33, p. 4].

3.11. Data quality management

Data quality management is an important practice in the MLOps workflow. According to Steidl et al. [7], the data processing stage includes the full lifecycle of working with data, including pre-processing, quality assurance, versioning and documentation [7, p. 21].

In Haller’s book [9, pp. 77-84], data quality management is considered in the context of monitoring and checking data in a production environment to ensure that the data used corresponds to reality.

In the data-centric MLOps lifecycle (figure 15), data quality management is performed at the following stages:

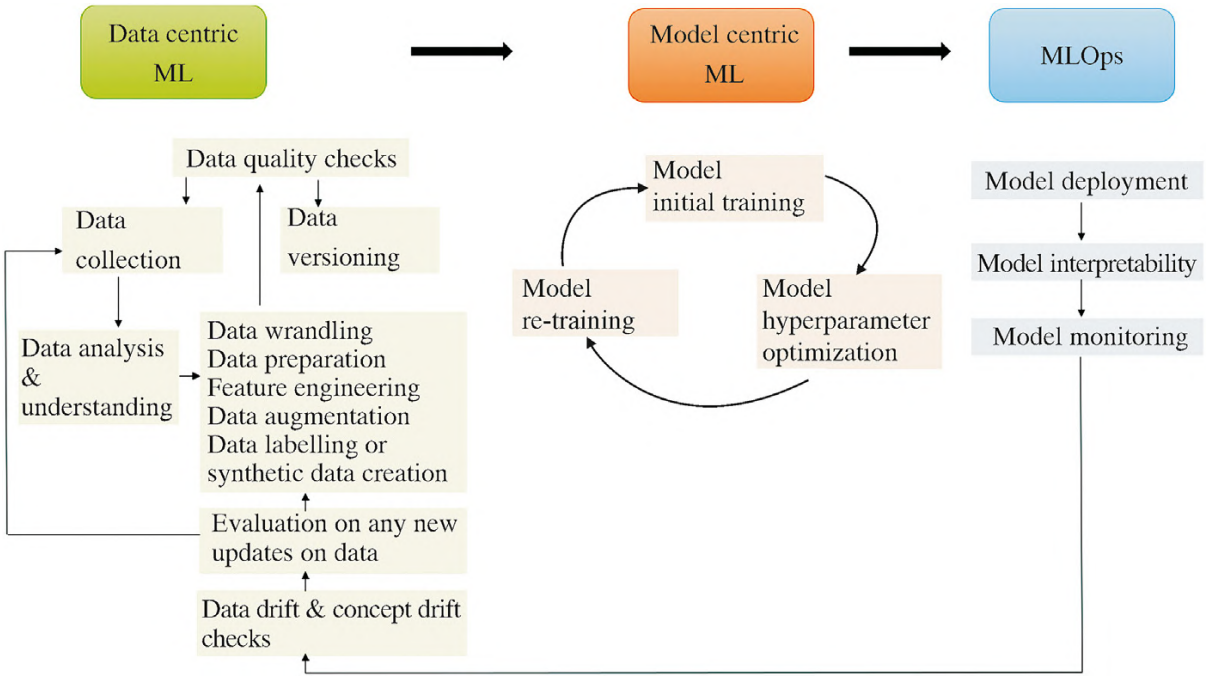

Figure 15: Data-centric MLOps lifecycle [26, p. 145].

1. Data collection – at this stage, data is created and obtained from various sources: data must be relevant, complete, consistent and reliable.
2. Data quality assessment and cleaning – collected data undergoes thorough quality checks using various metrics such as accuracy, completeness, consistency, timeliness, etc.; quality issues are identified and resolved – incorrect values, missing values, duplicates, noise; data cleaning techniques are applied to improve data quality.
3. Data augmentation and labeling – cleaned data is enriched with additional information and labeled according to the target task; at this stage, the quality of data labeling is also controlled to avoid errors and inaccuracies.
4. Data quality analysis – labeled data is analyzed for quality, its representativeness, class balance, presence of outliers and anomalies are checked; adjustments are made as needed to improve data quality.
5. Model training with quality control – based on quality data, ML model training takes place; during experiments, model and data quality metrics are tracked to ensure stability and reliability of results.
6. Model deployment with quality assurance – the model is deployed in the production environment only after thorough testing on quality test data; measures are taken to maintain data quality in the production environment.
7. Data and model quality monitoring – the deployed model and incoming data are constantly checked for quality: quality metrics, presence of anomalies in data, distribution shift are tracked; the model is retrained on new quality data as needed.

To guarantee data quality throughout the entire lifecycle of their use in MLOps, some approaches to data validation and verification need to be applied:

- data quality assessment by determining how suitable this data is for achieving business goals (completeness, uniqueness, integrity, validity, accuracy, timeliness) [26, pp. 2-3];

- data validation in single and cross batches by comparing data characteristics with the expected schema, as well as checking for data drift [7, p. 11];
- automated modular data tests based on a schema to identify errors in data and prevent them from entering the model training stage [7, p. 10];
- versioning of data and related artifacts (processing procedures, metadata, etc.) for traceability, reproducibility of results and compliance with regulatory requirements [7, p. 11];
- documenting data to an extent sufficient to ensure model verifiability [7, p. 12].

Timely detection and elimination of defects in data helps prevent wasting computational resources on low-quality data [7, p. 11].

In their work, Singh [26] present techniques for assessing the quality of "big data", which help identify datasets that may cause problems and unnecessary costs.

To implement data quality management practices, frameworks such as TensorFlow Extended (TFX) can be used, which have data validation components such as SchemaGen and ExampleValidator [7, p. 19].

3.12. Configuration management

Configuration files help create more robust software by moving all hardcoded variables into dedicated locations that can be split up or organized at the developer's discretion [34, p. 3].

Godwin and Melvin [34] propose a template that supports two types of configuration files – a config.py file that contains the template configuration and can include additional resources, such as databases and spreadsheets, and JSON files for storing specific program variables, such as thresholds or parameters. The config.py file is accessible through import statements, while JSON files are called from disk at runtime [34, pp. 3-4].

Yongqiang et al. [35] consider the use of a unified data model based on the YANG language for unifying the description of configuration data and simplifying their management. Neptune Labs [36] indicate that configuration management tools, such as Ansible (https://www.ansible.com/), Puppet (https://www.puppet.com/) and Chef (https://www.chef.io/), can be used to automate configuration and provisioning of MLOps platforms.

3.13. Model deployment strategies

Model deployment strategies:

- are implemented at the final stage of the MLOps process;
- require standardization, automation and encapsulation of models;
- are used for gradual transition of a new model to the production environment;
- rely on containerization.

Peltonen and Dias [28] highlight the following benefits of using containers for model deployment [28, p. 68]:

- model abstraction and process isolation by running multiple models in individual containers representing their runtime dependencies;
- ability to create model-specific containers that meet their packaging requirements;
- assignment to different processors (CPU, Mobile GPU, etc.);
- ability to allocate a separate container to facilitate post-processing functions;

- a model repository and container storage can be used for fast deployment;
- provides a means of standardized deployment;
- almost all existing continuous development pipelines facilitate model deployment in the form of containers;
- containers can be tailored to specific architectures of edge devices;
- Docker containers are a well-established industry standard;
- ease of rollback in case of failures.

Gunny et al. [37] describe in detail the development cycle and model deployment strategy using the DeepClean application as an example. A new version of the model is first deployed as a developer version, undergoes validation in conditions similar to production, and only then replaces the current model in the production environment. This uses a service architecture with NVIDIA Triton Inference Server, which supports simultaneous placement of the developer version and the production version of the model.

The authors identify two main approaches to model deployment [37, pp. 11-12].

In the traditional scenario, each user manages their own resources and model versions. Inconsistencies in libraries and dependencies, as well as model versions, lead to inconsistent results. Reduced computational requirements for inference lead to underutilization of hardware resources, depicted by green rectangles on each node (figure 16). More complex deployment scenarios require the use of multiple networks, exacerbating existing issues.

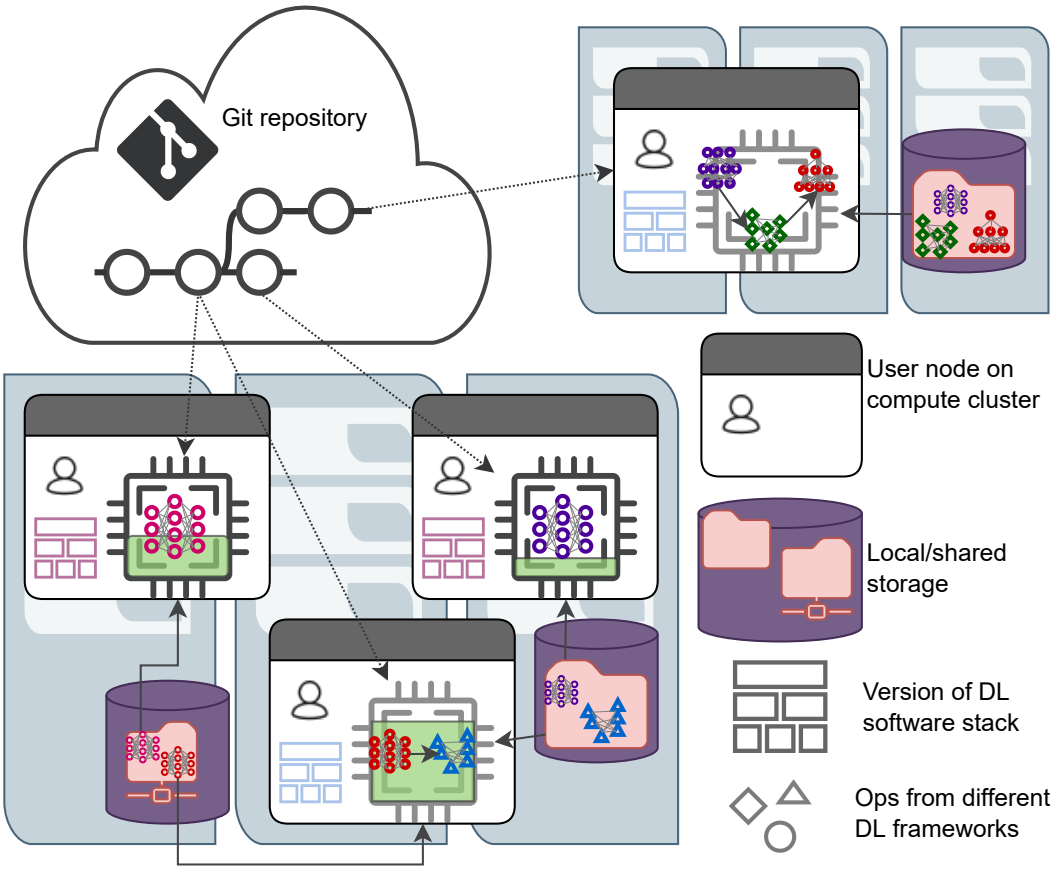

Figure 16: Traditional distributed deployment scenario [37, p. 12].

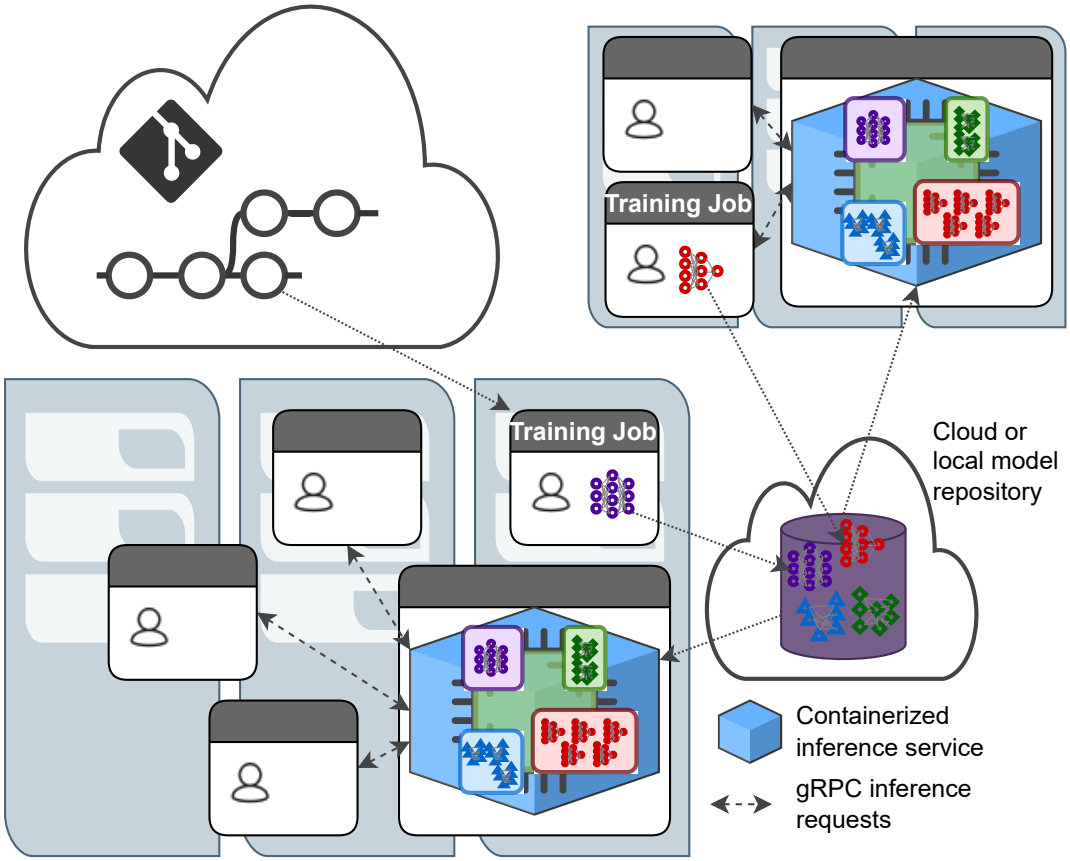

Figure 17: Deployment using the Inference-as-a-Service scenario [37, p. 12].

In the Inference-as-a-Service approach, a centralized service orchestrates models and provides unified interfaces for invoking models (figure 17). A centralized model store synchronizes all users and keeps them up to date. Pipelines send gRPC inference requests to the service using standardized APIs that abstract away the details of the inference execution itself. Inference is performed by a containerized service that can efficiently schedule asynchronous model execution, maximizing hardware compute capabilities in a portable and scalable manner. In this approach, containerization allows creating portable and isolated model execution environments.

Choosing the right model deployment strategy allows minimizing operational costs, ensuring consistent model operation for all users, and facilitating model version monitoring and control.

3.14. Infrastructure automation

Infrastructure as code is an important MLOps practice that allows treating infrastructure as code for its reliable and efficient deployment and management [38, p. 2]. Infrastructure automation relates to the deployment and monitoring stage in the MLOps process (figure 18). It ensures reliable creation of the necessary environment for deploying machine learning models and automates infrastructure tasks [38, p. 3].

Infrastructure as code involves describing infrastructure configuration in a declarative way in special files (for example, in YAML, JSON formats) or using special languages (for example, Terraform) or tools. These files describe the desired state of the infrastructure [38, p. 4].

Describing infrastructure settings as code allows automating the process of its creation, modification and management [38, p. 3]. This makes it possible to fully reproduce environments, quickly deploy resources and avoid manual setup errors [38, p. 4]. This approach is appropriate to use for creating complex dynamic infrastructures, when high repeatability and consistency of environments is needed,

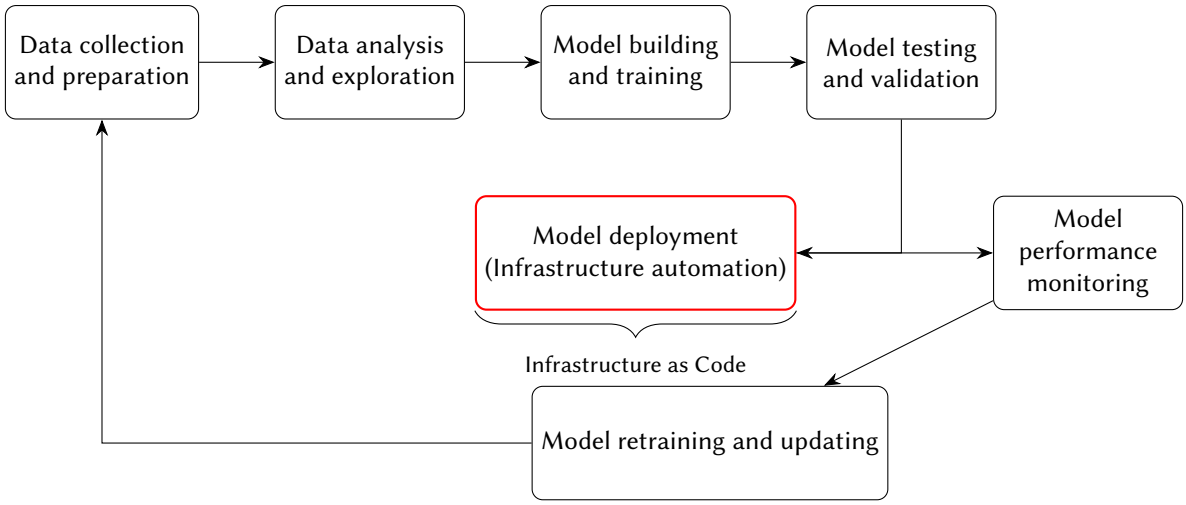

Figure 18: Infrastructure automation in the MLOps lifecycle.

to increase reliability and reduce time costs for manual configuration [38, pp. 2, 4].

The automated approach allows testing infrastructure as code, applying software development practices to it (code review, versioning, etc.). This contributes to improving stability and security, allows quickly tracking and resolving infrastructure issues [38, p. 4].

To implement Infrastructure as code in Azure DevOps, tools such as Azure Resource Manager (ARM) Templates, Terraform, Ansible, Chef are used [38, pp. 3-4]. They allow describing infrastructure as code and automatically creating or modifying it.

3.15. Collaboration and communication

Collaboration and communication between various stakeholders is a key MLOps practice for the successful implementation of machine learning and artificial intelligence projects in organizations. MLOps emphasizes how cross-functional teams, such as data analysts, system operators, as well as data and software engineers, collaborate through a harmonized process [7, p. 7].

The essence of the collaboration and communication practice lies in establishing effective interaction and information exchange between the various teams involved in the process of developing and implementing ML models – the data science team, development team, operations team, and business units. This practice is an important component of the development and implementation stage of the MLOps workflow.

As Kreuzberger et al. [2] point out, MLOps involves close collaboration between data science (machine learning) teams, which are engaged in data preparation and model development, software development engineers, who are responsible for integrating models into the production environment, and operations teams, which ensure the deployment and support of models [2, p. 2]. Effective communication is necessary at all stages of the ML model lifecycle, from defining business goals to monitoring models in the production environment. Without well-established collaboration, the development of ML solutions can be delayed, conflicts of interest and misunderstandings may arise between participants [39, p. 3].

The diagram (figure 19) shows the main participants in the MLOps process and the directions of their interaction:

1. The Data Science team prepares data and develops machine learning models.
2. The Development team integrates the developed models into software products.
3. The Operations team deploys models in the production environment and provides their support.
4. The Business team defines goals and requirements for ML solutions, and also receives results from the Data Science and Operations teams.

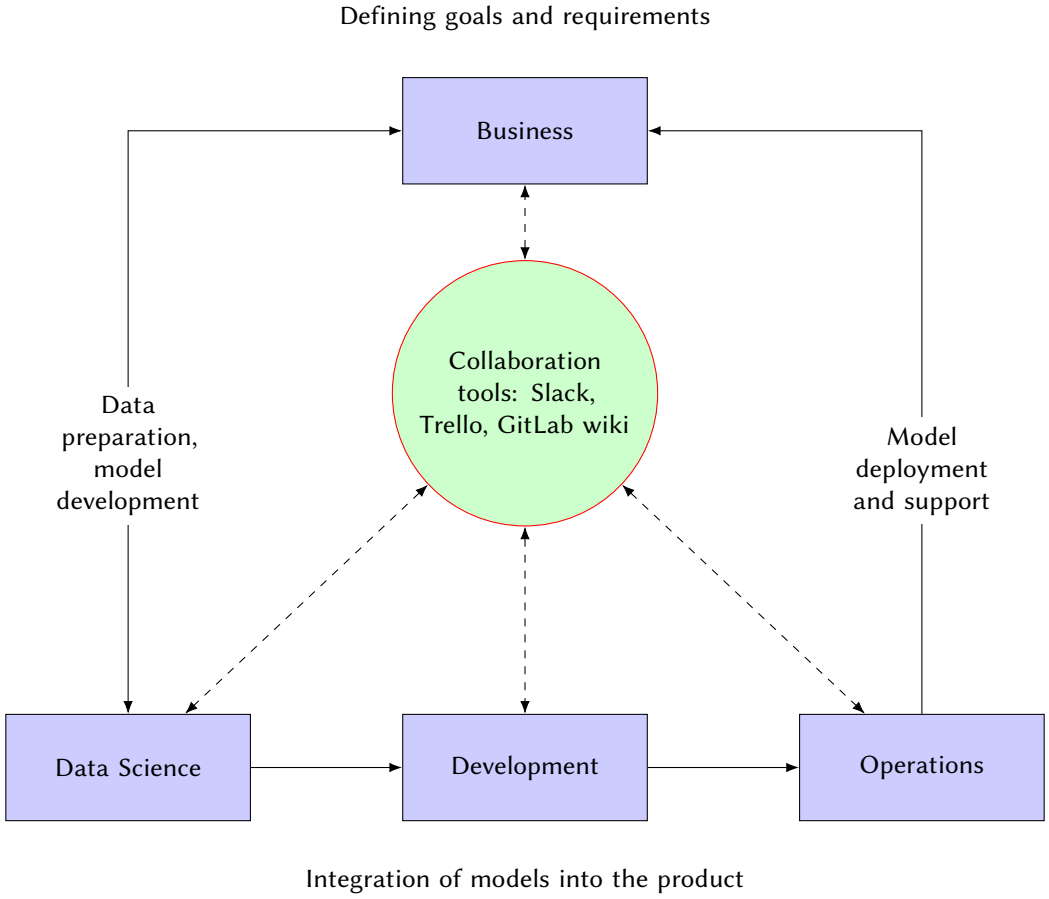

Figure 19: Diagram of collaboration and communication in MLOps.

To ensure effective communication in MLOps practice, the following approaches and tools are used [2, p. 4]:

- use of collaboration and knowledge sharing tools, such as Slack, Trello, GitLab wiki;
- regular meetings between teams to discuss status, problems and plans;
- clear definition of roles and areas of responsibility of process participants;
- use of version control systems (Git) for collaboration on code and models;
- automation of CI/CD processes to ensure transparency and reproducibility of development.

3.16. Risk management and compliance

Since ML models often make important decisions that affect people, risk management and compliance is a critically important MLOps practice, the essence of which is to ensure compliance of developed ML models and systems with regulatory requirements and standards (compliance) and manage potential risks associated with their development and operation.

This practice has a cross-cutting nature and manifests itself at different stages of the ML model lifecycle. Risk management in the context of MLOps involves identifying and mitigating potential risks associated with ML models, such as data bias, privacy breaches, model accuracy deterioration over time, etc. Compliance means ensuring that models comply with regulatory requirements, for example, regarding data protection [7, pp. 18-19]. The main condition for using this practice is the presence of regulatory requirements or industry standards that the ML system must comply with. Examples can be GDPR requirements for personal data protection or certifications in the healthcare industry [7, p. 5].

Steidl et al. [7] indicate that compliance aspects should be considered during data preparation, model training and validation, as well as deployment and monitoring [7, p. 18].

Methods of using this practice include regular data quality checks, testing models for bias and discrimination, implementing access control and data encryption, documenting model architecture and development process, monitoring model performance after deployment [7, pp. 11-12]. Yes, compliance is achieved by [7, p. 18]:

- quality control and origin of data used for model training to avoid violation of regulatory requirements;
- documenting and versioning models and data to ensure reproducibility of results and auditing;
- verifying models integrated into the production environment for compliance with requirements;
- continuous monitoring of deployed models for timely detection of potential violations or incorrect behavior.

The leading tools for ensuring risk management and compliance practice are version control systems for tracking changes in data and model code, testing tools for identifying problems in models, monitoring systems for tracking model accuracy in real time [7, pp. 12, 14]. At the same time, interviews with practitioners conducted by Steidl et al. [7] revealed that ensuring compliance of ML systems in various domains (for example, in healthcare) is a serious challenge due to the lack of established methodologies and software tools. At the same time, achieving compliance is a mandatory condition for obtaining the necessary certifications and permits from regulators.

The diagram in figure 20 shows three main MLOps stages: working with training data, model training, and system deployment (operationalization). At each stage there are blocks that indicate potential risks and compliance measures. This diagram illustrates the cross-cutting nature of risk management and compliance practice in MLOps, showing its presence and interconnections at each stage of the machine learning model lifecycle.

Figure 21 shows the relationships between the key principles, deployment process, and main MLOps practices that are applied at the stage of machine learning model deployment. The principles of automation and reproducibility influence the model deployment process, which is associated with the following MLOps practices: CI/CD, model deployment, data security and privacy, configuration management, model deployment strategies, and infrastructure automation.

4. Conclusions

A meta-synthesis of systematic reviews [6, 8, 5] and a review of products and providers [15] was performed in order to generalize knowledge about the implementation of MLOps practices for the effective deployment of machine learning models. The main conclusions obtained as a result of the meta-synthesis are as follows:

1. MLOps is an approach for managing, automating, and operationalizing the processes of developing, deploying, and supporting machine learning models based on practices from software engineering and DevOps. MLOps is based on a set of principles, processes, and practices that ensure effective development, deployment, and support of machine learning models.
2. The main stages of the MLOps lifecycle include the following processes: data collection and processing, model development and training, deployment, monitoring, and retraining of models.
3. Various frameworks and architectures are used to implement MLOps, such as open source platforms (MLflow, Kubeflow, TensorFlow Extended), cloud computing platforms (AWS, Google Cloud, Azure), containerization (Docker), and container orchestration (Kubernetes).

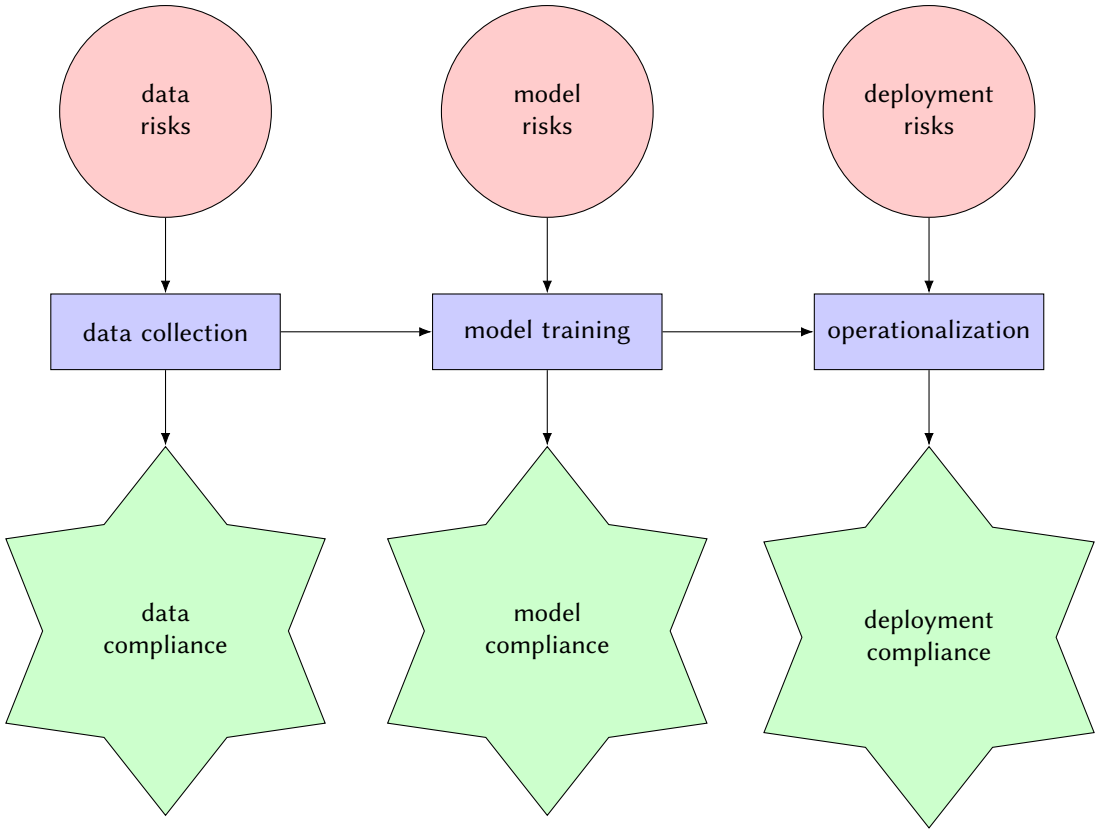

Figure 20: Diagram of risk management and compliance practice in MLOps.

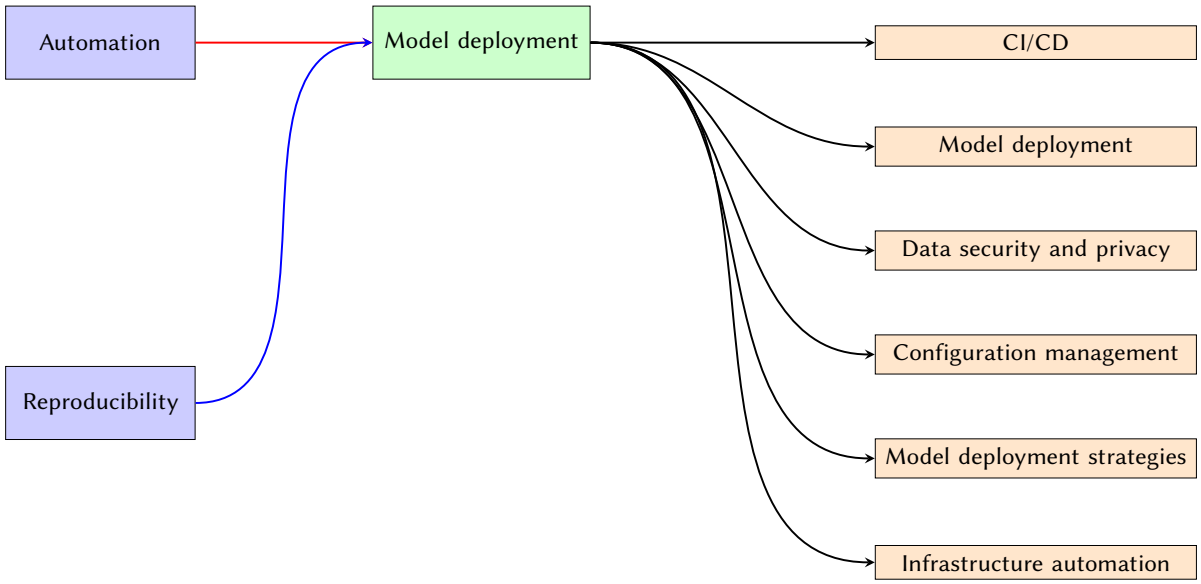

Figure 21: Diagram of relationships between principles, processes and MLOps practices for model deployment.

4. MLOps tools provide a wide range of features to support the machine learning model lifecycle, with a focus on automation, experiment tracking, versioning, monitoring, and model deployment.

5. The most common ways to deploy machine learning models in production environments are the use of container technologies, cloud platforms and services, and the deployment of models as web services.

6. Adapted software development maturity models, such as CMM, can be used to assess the maturity level of MLOps processes in organizations.

7. Successful MLOps implementation requires the involvement of specialists from different areas - software development, data engineering, machine learning, subject matter experts, and management.

8. The main challenges when deploying machine learning models in production environments are managing the model lifecycle, ensuring scalability and performance, monitoring and maintaining models in real-world conditions.

9. Open issues and challenges in MLOps are the need to develop standards and best practices, ensure interpretability and responsible use of models, effectively manage data, integrate knowledge from different fields.

10. The main opportunities and development trends of MLOps are the creation of standardized platforms, application in the context of distributed learning, and integration with other approaches to managing the lifecycle of data and models. Current and future areas of MLOps application include a wide range of industries, from finance and healthcare to IoT and natural language processing.

The conducted meta-synthesis showed that MLOps is a promising approach for the effective deployment of machine learning models in production environments, which requires further research and development to address existing challenges and realize potential opportunities.

Next, the key MLOps practices necessary for effective deployment of machine learning models were analyzed. The main conclusions obtained as a result of the analysis are as follows:

1. MLOps is based on a set of principles, processes and practices that ensure effective development, deployment and maintenance of machine learning models. The key principles of MLOps are automation, reproducibility, collaboration, continuous learning and data governance.

2. The main MLOps practices include: continuous integration and delivery (CI/CD), model and data versioning, ML pipeline automation, model performance monitoring, experiment management, model deployment and lifecycle management.

3. Additional MLOps practices, such as data security and privacy, model explainability and interpretability, data quality management, configuration management, model deployment strategies, infrastructure automation, collaboration and communication, risk management and compliance, are important for ensuring reliability, compliance with requirements and efficiency of MLOps processes.

4. The application of MLOps practices allows automating and standardizing the processes of development, deployment and maintenance of machine learning models, which increases the efficiency and reliability of ML solutions in the production environment.

5. Successful implementation of MLOps practices requires the use of appropriate tools and platforms, such as experiment management systems, data and model versioning, infrastructure automation and monitoring tools, as well as establishing effective collaboration between the different roles and teams involved in the process of developing and implementing machine learning models.

The analysis showed that the application of MLOps practices is critically important for the successful deployment of machine learning models in production environments. The implementation of these practices allows increasing the efficiency, reliability and reproducibility of the processes of development and operation of ML solutions, which is a necessary condition for their successful use in real business problems.

As a result of the comprehensive study, the MLOps practices necessary for the effective deployment of machine learning models were identified and analyzed. The main conclusions obtained during the study are as follows:

1. A meta-synthesis of systematic reviews was performed to generalize knowledge about MLOps practices. The conducted meta-synthesis showed that MLOps is a promising approach for the effective deployment of machine learning models in production environments, which requires further research and development to solve existing challenges and realize potential opportunities.

2. A diagram of relationships between MLOps principles, processes and practices is proposed. This diagram illustrates the interconnections between the key principles, stages of the machine learning model development and implementation process, as well as the main MLOps practices that are applied at each stage.

3. The most effective MLOps practices for model deployment have been identified, which include: continuous integration and delivery (CI/CD), model and data versioning, ML pipeline automation, model performance monitoring, experiment management, model deployment and lifecycle management, data security and privacy, model explainability and interpretability, data quality management, configuration management, model deployment strategies, infrastructure automation, collaboration and communication, risk management and compliance.

The obtained results have both theoretical and practical significance. The theoretical significance lies in the generalization and systematization of knowledge about MLOps practices necessary for the effective deployment of machine learning models. The practical significance of the obtained results lies in the possibility of their use by organizations for the implementation or improvement of MLOps processes in order to increase the efficiency and reliability of machine learning model deployment in production environments.

Further research may be aimed at developing detailed recommendations for the implementation of individual MLOps practices in organizations, creating new tools and platforms for automating and managing the lifecycle of machine learning models, as well as studying the effectiveness of applying MLOps practices in various industries and areas of machine learning model application.

Declaration on Generative AI: During the preparation of this work, the authors used Claude 3 Opus in order to: Text Translation, Abstract drafting. After using this service, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] P. V. Zahorodko, S. O. Semerikov, V. N. Soloviev, A. M. Striuk, M. I. Striuk, H. M. Shalatska, Comparisons of performance between quantum-enhanced and classical machine learning algorithms on the IBM Quantum Experience, Journal of Physics: Conference Series 1840 (2021) 012021. doi:10.1088/1742-6596/1840/1/012021.

[2] D. Kreuzberger, N. Kühl, S. Hirschl, Machine Learning Operations (MLOps): Overview, Definition, and Architecture, IEEE Access 11 (2023) 31866–31879. doi:10.1109/ACCESS.2023.3262138.

[3] G. Symeonidis, E. Nerantzis, A. Kazakis, G. A. Papakostas, MLOps - Definitions, Tools and Challenges, in: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022, pp. 0453–0460. doi:10.1109/CCWC54503.2022.9720902.

[4] M. Testi, M. Ballabio, E. Frontoni, G. Iannello, S. Moccia, P. Soda, G. Vessio, MLOps: A Taxonomy and a Methodology, IEEE Access 10 (2022) 63606–63618. doi:10.1109/ACCESS.2022.3181730.

[5] J. Diaz-de Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, A. Almeida, A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey, ACM Comput. Surv. 56 (2023) 84. doi:10.1145/3625289.

[6] G. Recupito, F. Pecorelli, G. Catolino, S. Moreschini, D. D. Nucci, F. Palomba, D. A. Tamburri, A Multivocal Literature Review of MLOps Tools and Features, in: 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2022, pp. 84–91. doi:10.1109/SEAA56994.2022.00021.

[7] M. Steidl, M. Felderer, R. Ramler, The pipeline for the continuous development of artificial intelligence models—current state of research and practice, Journal of Systems and Software 199 (2023) 111615. doi:10.1016/j.jss.2023.111615.

[8] A. Lima, L. Monteiro, A. P. Furtado, MLOps: Practices, Maturity Models, Roles, Tools, and Challenges – A Systematic Literature Review, in: Proceedings of the 24th International Conference on Enterprise Information Systems - Volume 1: ICEIS, INSTICC, SciTePress, 2022, pp. 308–320. doi:10.5220/0010997300003179.

[9] K. Haller, Managing AI in the enterprise: Succeeding with AI projects and MLOps to build sustainable AI organizations, Apress Berkeley, CA, 2022. doi:10.1007/978-1-4842-7824-6.

[10] E. e Oliveira, M. Rodrigues, J. P. Pereira, A. M. Lopes, I. I. Mestric, S. Bjelogrlic, Unlabeled learning algorithms and operations: overview and future trends in defense sector, Artificial Intelligence Review 57 (2024) 66. doi:10.1007/s10462-023-10692-0.

[11] A. B. Kolltveit, J. Li, Operationalizing machine learning models: a systematic literature review, in: Proceedings of the 1st Workshop on Software Engineering for Responsible AI, SE4RAI '22, Association for Computing Machinery, New York, NY, USA, 2023, p. 1–8. doi:10.1145/3526073.3527584.

[12] F. Calefato, F. Lanubile, L. Quaranta, A Preliminary Investigation of MLOps Practices in GitHub, in: Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 283–288. doi:10.1145/3544902.3546636.

[13] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, D. Moher, The PRISMA 2020 statement: an updated guideline for reporting systematic reviews, BMJ 372 (2021) n71. doi:10.1136/bmj.n71.

[14] C. Haertel, D. Staegemann, C. Daase, M. Pohl, A. Nahhas, K. Turowski, MLOps in Data Science Projects: A Review, in: 2023 IEEE International Conference on Big Data (BigData), 2023, pp. 2396–2404. doi:10.1109/BigData59044.2023.10386139.

[15] R. Cohen, Digital Strategy, Machine Learning, and Industry Survey of MLOps, in: Digital Strategies and Organizational Transformation, 2023, pp. 137–150. URL: https://tinyurl.com/33z6zpd3. doi:10.1142/9789811271984_0008.

[16] T. A. Sipe, W. L. Curlette, A meta-synthesis of factors related to educational achievement: a methodological approach to summarizing and synthesizing meta-analyses, International Journal of Educational Research 25 (1996) 583–698. doi:10.1016/S0883-0355(96)80001-2.

[17] J. Chrastina, Meta-synthesis of qualitative studies: background, methodology and applications, in: NORDSCI Conference proceedings, volume 1 of *NORDSCI Conference*, Saima Consult Ltd, 2018. doi:10.32008/nordsci2018/b1/v1/13.

[18] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, T. Zimmermann, 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, pp. 291–300. doi:10.1109/ICSE-SEIP.2019.00042.

[19] S. Dhanorkar, C. T. Wolf, K. Qian, A. Xu, L. Popa, Y. Li, Who needs to know what, when?: Broadening the Explainable AI (XAI) Design Space by Looking at Explanations Across the AI Lifecycle, in: Proceedings of the 2021 ACM Designing Interactive Systems Conference, DIS '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 1591–1602. doi:10.1145/3461778.3462131.

[20] L. E. Lwakatare, I. Crnkovic, J. Bosch, DevOps for AI – Challenges in Development of AI-enabled Applications, in: 2020 International Conference on Software, Telecommunications and Computer

Networks (SoftCOM), 2020, pp. 1–6. doi:10.23919/SoftCOM50211.2020.9238323.

[21] R. Akkiraju, V. Sinha, A. Xu, J. Mahmud, P. Gundecha, Z. Liu, X. Liu, J. Schumacher, Characterizing Machine Learning Processes: A Maturity Framework, in: D. Fahland, C. Ghidini, J. Becker, M. Dumas (Eds.), Business Process Management, volume 12168 of *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2020, pp. 17–31. doi:10.1007/978-3-030-58666-9_2.

[22] C. Min, A. Mathur, U. G. Acer, A. Montanari, F. Kawsar, SensiX++: Bringing MLOps and Multi-tenant Model Serving to Sensory Edge Devices, ACM Trans. Embed. Comput. Syst. 22 (2023) 98. URL: https://doi.org/10.1145/3617507. doi:10.1145/3617507.

[23] F. Bachinger, J. Zenisek, M. Affenzeller, Automated Machine Learning for Industrial Applications – Challenges and Opportunities, Procedia Computer Science 232 (2024) 1701–1710. doi:10.1016/j.procs.2024.01.168.

[24] K. Filippou, G. Aifantis, G. A. Papakostas, G. E. Tsekouras, Structure Learning and Hyperparameter Optimization Using an Automated Machine Learning (AutoML) Pipeline, Information 14 (2023) 232. doi:10.3390/info14040232.

[25] A. Bodor, M. Hnida, N. Daoudi, Machine Learning Models Monitoring in MLOps Context: Metrics and Tools, International Journal of Interactive Mobile Technologies (iJIM) 17 (2023) pp. 125–139. doi:10.3991/ijim.v17i23.43479.

[26] P. Singh, Systematic review of data-centric approaches in artificial intelligence and machine learning, Data Science and Management 6 (2023) 144–157. doi:10.1016/j.dsm.2023.06.001.

[27] J. Czakon, K. Kluge, ML Experiment Tracking: What It Is, Why It Matters, and How to Implement It, 2024. URL: https://neptune.ai/blog/ml-experiment-tracking.

[28] E. Peltonen, S. Dias, LinkEdge: Open-sourced MLOps Integration with IoT Edge, in: Proceedings of the 3rd Eclipse Security, AI, Architecture and Modelling Conference on Cloud to Edge Continuum, ESAAM '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 67–76. doi:10.1145/3624486.3624496.

[29] L. A. Melgar, D. Dao, S. Gan, N. M. Gürel, N. Hollenstein, J. Jiang, B. Karlas, T. Lemmin, T. Li, Y. Li, S. X. Rao, J. Rausch, C. Renggli, L. Rimanic, M. Weber, S. Zhang, Z. Zhao, K. Schawinski, W. Wu, C. Zhang, Ease.ML: A Lifecycle Management System for MLDev and MLOps, in: 11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings, 2021. URL: https://www.cidrdb.org/cidr2021/papers/cidr2021_paper26.pdf.

[30] H. Chen, M. A. Babar, Security for Machine Learning-based Software Systems: A Survey of Threats, Practices, and Challenges, ACM Comput. Surv. 56 (2024) 151. doi:10.1145/3638531.

[31] N. K. Gopalakrishna, D. Anandayuvaraj, A. Detti, F. L. Bland, S. Rahaman, J. C. Davis, "If security is required": engineering and security practices for machine learning-based IoT devices, in: Proceedings of the 4th International Workshop on Software Engineering Research and Practice for the IoT, SERP4IoT '22, Association for Computing Machinery, New York, NY, USA, 2023, p. 1–8. doi:10.1145/3528227.3528565.

[32] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1–38. doi:10.1016/j.artint.2018.07.007.

[33] F. Rezazadeh, H. Chergui, L. Alonso, C. Verikoukis, SliceOps: Explainable MLOps for Streamlined Automation-Native 6G Networks, IEEE Wireless Communications 31 (2024) 224–230. doi:10.1109/MWC.007.2300144.

[34] R. C. Godwin, R. L. Melvin, Toward efficient data science: A comprehensive MLOps template for collaborative code development and automation, SoftwareX 26 (2024) 101723. doi:10.1016/j.softx.2024.101723.

[35] D. Yongqiang, W. Xin, L. Yongbo, Y. Wang, Building Network Domain Knowledge Graph from Heterogeneous YANG Models, Journal of Computer Research and Development 57 (2020) 699–708. doi:10.7544/issn1000-1239.2020.20190882.

[36] Neptune Labs, MLOps Landscape in 2024: Top Tools and Platforms, 2024. URL: https://neptune.ai/blog/mlops-tools-platforms-landscape.

[37] A. Gunny, D. Rankin, P. Harris, E. Katsavounidis, E. Marx, M. Saleem, M. Coughlin, W. Benoit, A

Software Ecosystem for Deploying Deep Learning in Gravitational Wave Physics, in: Proceedings of the 12th Workshop on AI and Scientific Computing at Scale Using Flexible Computing Infrastructures, FlexScience '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 9–17. doi:10.1145/3526058.3535454.

[38] C. Vuppalapati, A. Ilapakurti, K. Chillara, S. Kedari, V. Mamidi, Automating Tiny ML Intelligent Sensors DevOPS Using Microsoft Azure, in: 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 2375–2384. doi:10.1109/BigData50022.2020.9377755.

[39] R. Sothilingam, V. Pant, E. S. K. Yu, Using i* to Analyze Collaboration Challenges in MLOps Project Teams, in: A. Maté, T. Li, E. J. T. Gonçalves (Eds.), Proceedings of the 15th International iStar Workshop (iStar 2022) co-located with 41th International Conference on Conceptual Modeling (ER 2022), Virtual Event, Hyderabad, India, October 17, 2022, volume 3231 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 1–6. URL: https://ceur-ws.org/Vol-3231/iStar22_paper_1.pdf.

A. Using the large language model Claude 3 Sonnet for analyzing systematic reviews

The queries were created on 10.05.2024. A PDF file with the article text was added to each query. A report according to the submitted plan was expected from the chatbot. The queries consisted of universal and variable parts.

Universal part of the query:

```
Using the added review article file as a data source,
write a detailed report on it according to the following plan.
```

Variable parts of the query:

1. 1. Year of publication.
   2. Research objective (paper aim).
   3. Research questions.

2. 4. Information sources (databases, etc.)
   5. Inclusion criteria.
   6. Exclusion criteria.
   7. Quality criteria.

3. 8. MLOps definition (if any).
   9. MLOps workflow stages (if any).

4. 10. What frameworks and architectures facilitate MLOps?

5. 11. What MLOps tools can be used to build ML pipelines for Continuous Deployment? What tools are used in the activities for operationalizing machine learning models?

6. 12. What are the main features offered by MLOps tools?

7. 13. How are machine learning models deployed in production environments?

8. 14. What maturity models are used to assess the level of automation in deploying machine learning models?

9. 15. What roles and responsibilities are identified in the activities of operationalization of machine learning models?

10. 16. What challenges are encountered with regard to deploying machine learning models in production environments?

11. 17. What are the open issues, challenges and particularities in MLOps?
    18. What are the opportunities and future trends in MLOps? What are the current and future fields in which MLOps is thriving?

B. Results of systematic reviews analysis

Table B.1
Results of systematic review analysis [6, 8, 5].

Comparison object	Review by Recupito et al. [6]	Review by Lima et al. [8]	Review by Diaz-de Arcaya et al. [5]
Year of publication	2022	2022	2023
Research objective	To identify tools that allow creating MLOps pipelines for continuous deployment. To analyze the main characteristics and functions of these MLOps tools to provide a comprehensive overview of their value.	Review of existing literature to identify practices, standards, roles, maturity models, challenges and tools used to automate activities of operationalizing machine learning models into industrial operation (MLOps).	The main objective of this systematic literature review is to provide an understanding of the implementation of MLOps and AIOps methodologies in both industry and academia. The authors seek to highlight the challenges, opportunities and future trends in these areas.
Research questions	We answer the following main research question: What tools and capabilities enable developers to create ML-enabled software systems? Which was detailed into two sub-questions: RQ1. What MLOps tools can be used to create machine learning pipelines for continuous deployment? RQ2. What are the main features offered by MLOps tools?	RQ1: How are machine learning models deployed in production environments? RQ2: What maturity models are used to assess the level of automation in deploying machine learning models? RQ3: What roles and responsibilities are defined in the activities of operationalizing machine learning models? RQ4: What tools are used in the activities of operationalizing machine learning models? RQ5: What challenges are encountered when deploying machine learning models in production environments?	Q1: What are the open issues, challenges and particularities in MLOps and AIOps? Q2: What are the opportunities and future trends in MLOps? Q3: What are the opportunities and future trends in AIOps? Q4: What platforms and architectures facilitate MLOps and AIOps? Q5: What are the current and future fields in which MLOps and AIOps are thriving?

Continued on next page

Continuation of table B.1

Comparison object	**Review by Recupito et al. [6]**	**Review by Lima et al. [8]**	**Review by Diaz-de Arcaya et al. [5]**
Information sources	Google Scholar – for searching scientific literature, such as journals, books and dissertations. Google Search – for searching so-called "gray" literature, such as blog posts, developer sites, webinars, GitHub repositories and YouTube videos. Using both academic (white literature) and professional (gray literature) sources allowed the authors to explore MLOps from different perspectives – theoretical and practical.	The automated search was conducted in the following electronic research databases: ACM Digital Library, IEEE Xplore, ScienceDirect and SpringerLink.	To search for relevant articles, the authors used several databases and repositories, including arXiv, Springer, IEEE. However, the main source was the Scopus database from Elsevier, as it contains metadata and abstracts of many publications.
Inclusion criteria	The study discusses components of a minimal end-to-end MLOps workflow. The study discusses MLOps practice or machine learning-based applications. The study relates to the implementation of MLOps tool(s). The study describes experiences, opinions, or practices regarding MLOps pipelines.	Studies related to machine learning operations (MLOps) in general. Studies evaluating the lifecycle of machine learning solutions. Studies related to maturity models of the machine learning process. Studies analyzing roles and responsibilities involved in the development and deployment of machine learning solutions. Studies covering tools for deploying machine learning solutions. Studies identifying challenges for the development and deployment of machine learning models.	Articles published between 2018 and 2023, identified by search queries, containing new ideas and closely related to the topic of MLOps and AIOps were included in the analysis.
Exclusion criteria	The study does not provide details on the design or implementation of MLOps tool(s). The study only proposes the design of a certain component of machine learning pipelines. The study does not provide or reference details on machine learning automation. The study refers to commercial platforms that offer MLOps applications to promote their development and deployment services.	Studies not published in English. Studies related to the application of machine learning models. Short papers or posters. Studies not related to machine learning operations. Studies whose content is not accessible. Articles not relevant to the research questions.	Publications not in English, retracted publications, publications from irrelevant publishers, subscription materials without access, and articles with insufficient citations (depending on the year of publication) were excluded.

Continued on next page

Continuation of table B.1

Comparison object	Review by Recupito et al. [6]	Review by Lima et al. [8]	Review by Diaz-de Arcaya et al. [5]
Quality criteria	The repository must have at least 100 stars The YouTube video must have been viewed at least 1000 times	Does the study report unambiguous findings based on evidence and arguments? Does the study represent a research project and not an expert opinion? Is the context being analyzed fully described in the study? Are the research objectives clearly defined? Are the research results properly validated?	Comprehensive literature review and gap identification Verification of results on a use case Number of research questions addressed Publication under an open license Type of publication (journal/other) Publication in a high-impact journal Number of citations
MLOps definition (if available)	MLOps is a practice that helps model, develop and operate the machine learning lifecycle, drawing on DevOps principles and practices.	A set of practices and principles for operationalizing data science solutions, used to automate the deployment of machine learning models into an operational environment	MLOps uses machine learning, DevOps and data engineering to bring machine learning systems into production, facilitating the creation of machine products.
MLOps workflow stages (if available)	Data extraction for integration Data analysis Data cleaning, transformation and feature engineering to split data Model training Model validation to assess the quality Model deployment in target environments Model monitoring	Data collection Data transformation Continuous training Continuous model deployment Presentation of results Monitoring of machine learning solutions	Data management Distributed training Deployment Monitoring Retraining The need to manage the lifecycle and key components of AI applications using specialized platforms and tools is emphasized.
Frameworks and architectures that facilitate MLOps implementation	Continuous training pipelines deployed via CI/CD Orchestration platforms TensorFlow Extended (TFX) Machine learning cloud platforms	MLflow Kubeflow Polyaxon Comet.ml Kafka-ML MLModelCI	HPC Cloud computing Edge/IoT platforms Serverless architectures Frameworks for integration Automatic data labeling methods Proactive incident management Platforms for orchestration Semantically enhanced pipelines Architectures for distributed training and deployment Frameworks for monitoring Programming languages Containerized solutions AutoML software Use of APIs Deep learning and neural networks

Continued on next page

Continuation of table B.1

Comparison object	**Review by Recupito et al. [6]**	**Review by Lima et al. [8]**	**Review by Diaz-de Arcaya et al. [5]**
MLOps tools for creating machine learning pipelines and operationalizing models	Machine learning cloud platforms (AWS SageMaker, AzureML, Google AI Platform) Orchestration platforms (Apache Airflow, Jenkins, Kubeflow, MLflow, Polyaxon, Seldon Core, Valohai) Configuration frameworks (TensorFlow Extended, Gitlab)	MLflow - this open platform has components such as MLflow Projects and MLflow Model Registry Kubeflow Kafka-ML MLModelCI	Containerized solutions (e.g., Docker) Serverless computing (AWS Lambda, Azure Functions, etc.) Continuous integration/continuous deployment (CI/CD) tools Monitoring of processes and events MLOps automation using AutoML Containerization for packaging model dependencies. API technologies for deployment as a web service.
Main features offered by MLOps tools	Common features related to all phases of machine learning pipelines Data management features Model management features	Experiment tracking Model packaging and versioning ML project management Model registry Continuous integration and delivery Model monitoring Hyperparameter tuning Portability	Data, model, and code versioning Workflow orchestration and automation Monitoring of processes and events Integration with cloud and edge infrastructure Containerization MLOps automation using AutoML
Methods of deploying machine learning models in production environments	Machine learning cloud platforms (AWS SageMaker, AzureML, DotScience, Google AI Platform) Orchestration platforms (Apache Airflow, Jenkins, Kubeflow, MLflow, Polyaxon, Seldon Core, Valohai) TensorFlow Extended (TFX)	MLOps is considered as a set of practices and principles for operationalizing Some MLOps tools, such as MLflow, Kubeflow, and Kafka-ML The role of “Deployment Lead”	Deployment in the cloud Using cloud computing resources Providing isolation Hybrid approaches Deployment on edge/IoT devices Deployment directly on IoT devices and mobile devices. TensorFlow Lite and Core ML. Overcoming limitations Containerized deployments Packaging models Docker. Serverless architectures Deploying ML functions as a service Reducing costs Deployment via APIs

Continued on next page

Continuation of table B.1

Comparison object	**Review by Recupito et al. [6]**	**Review by Lima et al. [8]**	**Review by Diaz-de Arcaya et al. [5]**
Maturity models for assessing the level of automation in deploying machine learning models	Support for continuous integration and continuous deployment (CI/CD) The ability to automatically tune Full automation of model management processes	Maturity model proposed by Amershi et al. [18] Dhanorkar et al. [19] classify organizations into three levels of maturity Lwakatare et al. [20] describe five stages of improvement in development practices Akkiraju et al. [21] propose an adaptation of the Capability Maturity Model (CMM)	The article does not define specific maturity models for assessing the automation of machine learning model deployment, but emphasizes the importance of adapting software development practices to the field of machine learning.
Roles and responsibilities identified in the activities of operationalizing machine learning models	Data scientists – responsible for developing and training Data engineers – responsible for extracting, processing, transforming, and ensuring the quality of data DevOps engineers – responsible for automating deployment processes and managing operational environments Product managers and business stakeholders – provide requirements for models and participate in decision-making	Domain specialist – has deep knowledge of the subject area Computational scientist/engineer – has high technical skills ML scientist/engineer – responsible for designing Provenance specialist – manages the supply of data Manager – evaluates models Application developer – develops applications Deployment lead – assesses aspects	Software developers Data specialists/data scientists Operations engineers Domain experts Management/stakeholders
Challenges encountered when deploying machine learning models in production environments	Complexity of managing Ensuring consistency Integration of various tools Automation of all stages Monitoring model performance Scaling infrastructure Ensuring security	Integration of software development Implementation of MLOps/AIOps practices Machine learning models need monitoring Identifying infrastructure components Deploying and versioning	Managing the ML lifecycle Gap between software engineering Data management Distributed and parallel execution Diversity of computing infrastructure Monitoring Explainability
Challenges encountered when deploying machine learning models in production environments	Lack of standardization Ensuring portability Configuration and integration Full automation Understandability	Integration of software development processes Implementation of MLOps practices It is necessary to go beyond analyzing model prototypes Careful monitoring Determining infrastructure Addressing scalability Need for versioning Automation Managing the lifecycle Integration with DevOps	Lack of skilled personnel Data management issues Complexity of orchestration Heterogeneity of hardware Need for continuous monitoring Lack of explainability Scaling and performance issues

Continued on next page

Continuation of table B.1

Comparison object	**Review by Recupito et al. [6]**	**Review by Lima et al. [8]**	**Review by Diaz-de Arcaya et al. [5]**
Opportunities, future trends, and areas of application of MLOps	Standardization of MLOps practices and tools Improved support environments. Advances in automation and Integration of MLOps with DevOps and DevSecOps. Increased attention to data management Areas of application Financial services and banking Healthcare and biotechnology Manufacturing and Internet of Things Retail and e-commerce Telecommunications Transportation and logistics	Demand for MLOps tools and platforms is expected to grow Industries where MLOps is actively developing – finance, healthcare, industry, retail, transportation, and logistics. Possible development of specialized Integration of MLOps with DevSecOps, MLSecOps concepts Emergence of new roles and competencies	Involvement of business units Greater attention to the ML lifecycle Better data management practices Use of new hardware platforms Use of containers, serverless technologies Development of versioning tools Industries where MLOps is thriving: Traditional industries Innovative industries Academic disciplines Communication and networking technologies Healthcare Scientific activity