Institute of Software Engineering
Universitätsstraße 38
70569 Stuttgart
Germany

Master Thesis

Towards Addressing MLOps Pipeline Challenges: Practical Guidelines Based on a Multivocal Literature Review

Prathima Siddappa

Course of Study:	Computer Science
Examiner:	Prof. Dr. Stefan Wagner
Supervisor:	Dr. Justus Bogner
Commenced:	April 11, 2022
Completed:	October 11, 2022

Abstract

Machine learning and artificial intelligence have been adopted in many businesses recently. The adoption of continuous software engineering practices such as DevOps in deploying machine learning projects to production is termed as MLOps. However, not all ML projects reach production due to the various complexities involved. This paper presents the challenges present in different components of the MLOps pipeline namely the data manipulation pipeline, model creation pipeline, and deployment pipeline. We conduct a systematic literature review and grey literature review to identify the challenges in the MLOps pipeline. Based on this data, we synthesize practical and relevant guidelines for practitioners, e.g., enriched with available tool support that could be used to implement them. The applicability of a selection of these guidelines is then demonstrated qualitatively by using them in the context of an example case with industry-relevant ML components. The contribution of this paper is threefold. First, we review contemporary literature to provide an overview of the state-of-the-art in MLOps challenges, solutions, and tools used. Second, we present seven practical guidelines based on the review. Third, we apply these guidelines with practical demonstration. The results will provide insight into how certain MLOps challenges can be overcome by following guidelines (not tool specific) mentioned in our study in the area of research and industry.

Contents

List of Figures

List of Tables

Listings

1 Introduction

1.1 Context and Motivation

Machine learning and artificial intelligence are gaining importance nowadays due to their application in various fields [23]. They have a significant impact in understanding the trends in customer behavior, business operational patterns and decision-making process in companies [24] . ML applications are not explicitly programmed, instead, they are developed on learning from data [23]. Therefore, the deployment, operation, and maintenance of machine learning models differ significantly from traditional applications [35].

The lifecycle of machine learning applications involves data acquisition, model development, model deployment, and model audit [14]. The work needed to properly create and implement an AI-based software system includes much more than just creating ML models. A problem that has persisted since the development of the first model is the incorporation of machine learning models in production [4]. Years have been spent by data scientists, machine learning engineers, front-end engineers, and production engineers to collaborate and pool their knowledge to develop a mechanism to release models that are ready for production. It is difficult to get over the various obstacles in this endeavor. Because of this, only a small portion of ML initiatives succeed in getting to production [4].

To boost value creation and automate the end-to-end life cycle of ML, data scientists and operations teams are striving to adapt DevOps concepts to their ML systems in businesses and this is termed as MLOps [24]. MLOps are a group of techniques for deploying and maintaining machine learning models in production [2]. MLOps, like DevOps, handles the continuous training, testing, deployment, and monitoring of ml models [21]. However, the practice of MLOPs is still in its early stages [24]. Although some of the issues encountered when developing ML-based software systems can be resolved by modern methodologies, there are currently no established protocols for how to combine them with ML workflow processes in practice [23].

1.2 Objectives

The main objective of the thesis is to synthesize proposed guidelines for addressing common MLOps pipeline challenges. The guidelines should be relevant and actionable for practitioners, e.g., enriched with available tool support that could be used to implement them. As a starting point, a systematic literature review and a grey literature review on MLOps pipeline challenges and their solutions will provide the basis for this study [32]. Based on the results, practical guidelines are proposed. The applicability of a selection of these guidelines is then demonstrated qualitatively by using them in the context of an example case with industry-relevant ML components.

1.3 Structure of the Thesis

The thesis is structured in the following way. Chapter 2 includes the necessary foundations such as DevOps, ML, and MLOps. Chapter 3 provides the State-of-the-art and related studies that inspired the thesis. Chapter 4 describes the research design for the thesis. Chapter 5 presents the results of the study, and related analysis. It documents the MLOps challenges and solutions, and tools from the research analysis. It also proposes practical guidelines for these challenges. Chapter 6 describes the details of the practical implementation of the proposed guidelines. Chapter 7 discusses the results and limitations of our research. Chapter 8 summarizes the results of the thesis, and proposes future work.

2 Background

2.1 DevOps

Many software development teams are trying to shift their software development process from the traditional waterfall strategy to DevOps recently. DevOps is a collection of practices designed to speed up the process of pushing a change into production while maintaining high standards [22]. It is the combination of development and operation and they work together effectively by integrating processes, tools, and data [29] as represented in figure 2.1.

The practices mentioned in DevOps are Continuous Integration (a technique in which numerous contributors regularly combine their contributions to form a bigger whole), Continuous testing (proving a system's correctness automatically by running a set of automated tests whenever a change is made), Staged deployments, upgrade/downgrade testing (seeing whether a software upgrade is functional while also seeing if the product can be uninstalled if necessary), continuous deployment, Monitoring, Sharing, Infrastructure as Code (establishing a project's infrastructure as a block of code), infrastructure as a Service (employing an automated mechanism to request infrastructure, then charging for that infrastructure following how much use it receives) [12].

2.2 Machine Learning

Machine learning is a broad term which refers to computer techniques that uses experience to enhance performance or produce precise predictions [5]. Experience in this context refers to the previous knowledge that the learner has access to, which frequently takes the form of electronically gathered data that is made available for analysis [5]. Machine learning is broadly classified into three types. Supervised learning, Unsupervised learning, reinforcement learning [27].

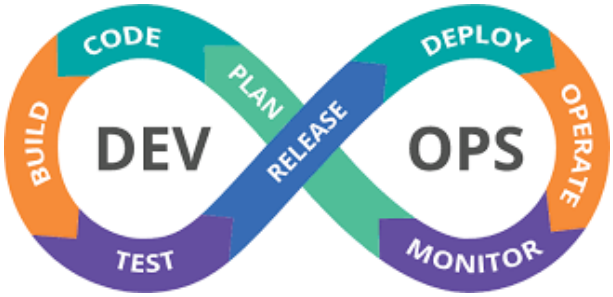

Figure 2.1: DevOps[1]

- Supervised learning: Based on labeled training data, supervised learning establishes the relationship between input and output. The machine learning model is used to predict the outcomes utilizing unlabeled data after being trained on labeled data. Eg: Image recognition,Speech recognition,Spam detection
- Unsupervised learning: Without labels or categorization, unsupervised learning use ML algorithms to find commonalities in the data. The goal is to show the model the structures of a dataset and then automatically classify new data. Eg: Anomaly/fraud detection, Gene clustering, Image segmentation
- Reinforcement learning: The effectiveness of the learning model is increased through reinforcement learning, which employs reward and punishment systems. Learning here is interactive with the environment and iterative. It comprises a learning phase called exploration, followed by a use phase called exploitation. Eg: Board and video game AI, Robotics,Traffic light control

Machine learning life-cycle consists of following stages [28].

- Data Collection: It focuses on obtaining data samples by observing and measuring the real-world system, process, or phenomenon for which an ML model needs to be constructed.
- Data Analysis: This step understands the schema of data and identifies the need for data preparation and preprocessing.
- Data Preprocessing: This step involved data cleaning, producing consistent data for training and validation, and engineering features to help in training (Feature engineering). Furthermore, data is labeled in the case of supervised learning.

[1] https://software.af.mil/training/devops/

- Model learning: It involves selecting the model depending on the problem type (e.g., classification or regression). The model is trained using training data until the error is minimized by providing a different value of parameters and hyperparameters. (Hyperparameter regulates important ML model properties including overfitting, underfitting, and model complexity).
- Model verification: The model's performance is assessed against the validation data set to ensure that the trained model performs well on unseen data (generalization).
- Model deployment: The validated model is deployed to production and monitored for its performance.

So, building an ML model is only a minor component of a comprehensive ML system, as shown in the figure 2.2.

2.3 MLOps

MLOps is "a software engineering approach in which an interoperable team produces machine learning applications based on code, data, and models in small, secure new versions that can be replicated and delivered reliably at any time, in short, custom cycles" [8][30]. It is the combination of DevOps and machine learning [14]. The concept of continuous integration and continuous delivery is applied in building machine learning systems. Although it appears straightforward, it is not. The reason behind this is that a machine learning model not only involves code but also data. Given that the data is dynamic, all the steps of ML workflow defined above have to be reproduced, ml models can become stale and there might be concept drift and data drift in production[3]. As a result, MLOps introduce a new practice Continuous Training (CT) along with CI and, CD which seeks to automatically update the model as necessary [14]. Therefore, it is evident that in comparison to DevOps, MLOps is significantly more complicated and has more techniques for models and data [26] [31] [24].

MLOps pipeline usually consists of a data manipulation pipeline, model creation pipeline, and deployment pipeline [14]. The data manipulation pipeline involves the automated collection of data from various sources, data cleaning, and data transformation. The model creation pipeline involves feature selection and model training. The deployment pipeline involves moving the model to production and monitoring.

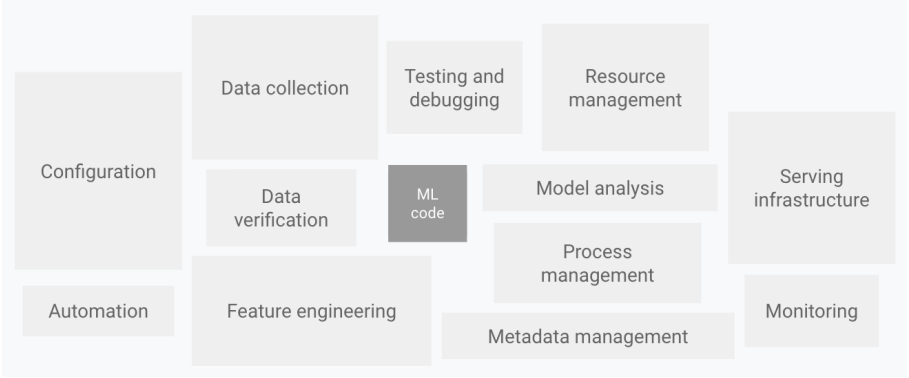

Figure 2.2: ML System[2]

[2]https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

3 Related Work

This chapter discusses the related work which is relevant for our study.

3.1 Data Quality and Data Management View of MLOps

In 2021, a study was published by Renggli et al. [6]. In this paper, they discuss that there is significant interdependence between the quality of the machine learning model and the quality of the data used to train or evaluate a model. They have described four of their earlier works, which have a common theme of providing capabilities to improve the MLOps process. Each work addresses a particular issue with MLOps and poses technical problems in analyzing downstream ML operations, data quality, and limitations. The main observation is that data management issues frequently accompany MLOps concerns and there is a high correlation between the quality of data and the quality of machine learning models. Understanding, assessing and enhancing the quality of ML models frequently depends on understanding, identifying, evaluating, and fixing the underlying data quality problems.

Similarly, as part of Google research, a paper by Neoklis et al. was published [25]. It examines problems with data management that occur in the environment in which production-ready machine learning pipelines are used. Authors using their own experience with such massive pipelines as a guide, concentrated on difficulties connected to understanding, validating, cleaning, and enriching training data.

3.2 Problems in Development of AI Applications

One of the few papers discussing the challenges is by Lucy et al. [23]. This study is based on a systematic literature review and empirical evidence gathered by practitioners from companies when conducting research. The difficulties in creating complicated systems with ML components are highlighted in this work, and potential solutions combining DevOps and

ML workflow processes are discussed. To demonstrate these difficulties and the potential solutions, industrial cases are offered.

Similar to Lucy, another paper by Georgios et al. briefly discusses this topic [14]. The main aim is to define the MLOps life-cycle and the maturity levels that define the level of automation in the MLOps pipeline. It presents various tools that help in different stages of ML workflows and highlights the difficulties in choosing the right tool. It discusses how MLOps and AutoML together can help to build better pipelines. They conclude by saying a fully developed MLOps system that receives continuous training can lead us to ML models that are more realistic and effective.

Janis Klaise et al. examined model monitoring and explainability of models in their paper published in 2020 [19]. It describes the importance of monitoring to maintain the accuracy of the model in production. It identifies the key areas for the success of ML application such as monitoring model performance and metrics, detecting outliers and drift, and explaining model predictions. The challenges in these areas are discussed along with the successful implementation of production-ready solutions using open source solutions.

3.3 Best Practices for ML Engineering

Certain papers describe the best practices of software engineering that can be employed in AI Engineering. One such paper is by Serban et al. [1]. The main goal is to assess the current state of the art in software development, deployment, and maintenance teams that incorporate ML components. Firstly they mine academic and grey literature to identify 29 best software engineering practices of which some were new and some were adopted from SE to design ML applications. To ascertain the extent of adoption of these procedures and to validate their alleged effects, they surveyed 313 practitioners. Using various statistical models, they also assessed correlations and looked into linear and non-linear links between practices and their reported effects.

3.4 Summary

While conducting our study on this subject, we have discovered that there is a lack of research on MLOps challenges and solutions in various components of the MLOps pipeline. This increases the current demand for our study.

In particular, the first two papers in section 3.1 are related to data challenges and they only speak about how the quality of data, dirty data, and validating data affect the data model. However, it does not cover all the issues faced in the data manipulation pipeline, for example, difficulty in accessing the data, and shortage of diverse samples. In section 3.2, the first two papers speak about the challenges and solutions briefly and the last paper speaks about monitoring and explainability. The papers in this section discuss a few deployment pipeline challenges. In section 3.3, the paper discusses the best SE practices for AI engineering but does not provide any set of directions for practitioners in MLOps.

None of the papers discusses the challenges in the model creation pipeline and proposes any practical guidelines for the practitioners. This thesis attempts to shed light on various challenges faced in the data manipulation pipeline, model creation pipeline, and deployment pipeline and proposes practical guidelines for practitioners.

4 Research Methodology

This chapter focuses on the research questions for our study in the first section. The next section describes the motivation and overview of our multivocal literature review and presents the protocols for our grey literature review and systematic literature review.

4.1 Research Questions

In accordance with the objective of our study, we came up with the following research questions:

- **RQ1:** What are the common technical challenges faced in different components of MLOps pipelines?
- **RQ2:** What are the proposed solutions for MLOps pipeline challenges?
- **RQ3:** Based on the results of RQ1 and RQ2, What are relevant and actionable MLOps guidelines for practitioners?
- **RQ4:** For which of the guidelines can be practical applicability demonstrated using industry relevant ML components?

The motivation for the first research question is that an MLOps pipeline usually consists of a data manipulation pipeline, model creation pipeline, and deployment pipeline [14]. Identifying the technical challenges in each component helps to overcome the difficulties individually. For the second research question, the focus is on finding the solutions for these technical challenges. The third research question proposes practical guidelines which are relevant and actionable for practitioners, e.g., enriched with available tool support that could be used to implement them. Finally, the last research question focuses on which practical guidelines can be demonstrated using industry relevant components.

We considered Multivocal Literature Review to be the most appropriate method to answer the research questions (RQ1 and RQ2).

4.2 Multivocal Literature Review

This section reports our motivation for selecting Multivocal Literature Review (MLR) as the most proper method to answer the research questions (RQ1 and RQ2). Section 4.2.2 presents an overview of general methodology of MLR. The last two sections present the protocols for Grey Literature Review (GLR) and Systematic Literature Review (SLR).

4.2.1 Motivation

"A literature review is an objective, thorough summary and critical analysis of the relevant available formal and non-formal literature on the topic being studied" [7]. For our study, we consider both research and non-research literature. MLOps is a relatively new area, and there are not many review papers on it [14]. Along with scientific literature, considering grey literature helps reduce the gap between academic research and professional practice [15]. It also helps us to keep up with current practices in the industry. There is emerging research going on for this topic, so considering both SLR and GLR gives the best of both. Therefore, we opt for a multivocal literature review. A multivocal literature review is a type of systematic literature review that incorporates both published (formal) literature (e.g., journal and conference papers) as well as grey literature (such as blog postings, videos, and white papers) [32]. MLR identifies the actual requirements in research study and industrial settings in line with our research objectives and is the best way to respond to our four research questions.

RQ1 and RQ2: MLR allows us to answer these two questions because it not only mentions the challenges and solutions from academic study but also the difficulties faced by practitioners on daily basis in building MLOps pipeline and gives insight into the state-of-the-practice in the industry.

RQ3 and RQ4: MLR allows us to combine the best practices for building MLOps pipeline from both scholarly research and industry practices. It mentions the available open source tools and platform-specific tools for various functions in MLOps pipeline.

4.2.2 Overview of Multivocal Literature Review

Figure 4.1 presents the overview of multivocal literature review. The first step in conducting the MLR was to establish the need for MLR. The motivation behind choosing MLR for our study is covered in section 4.2.1. The next step is to clearly state the goals of MLR. In

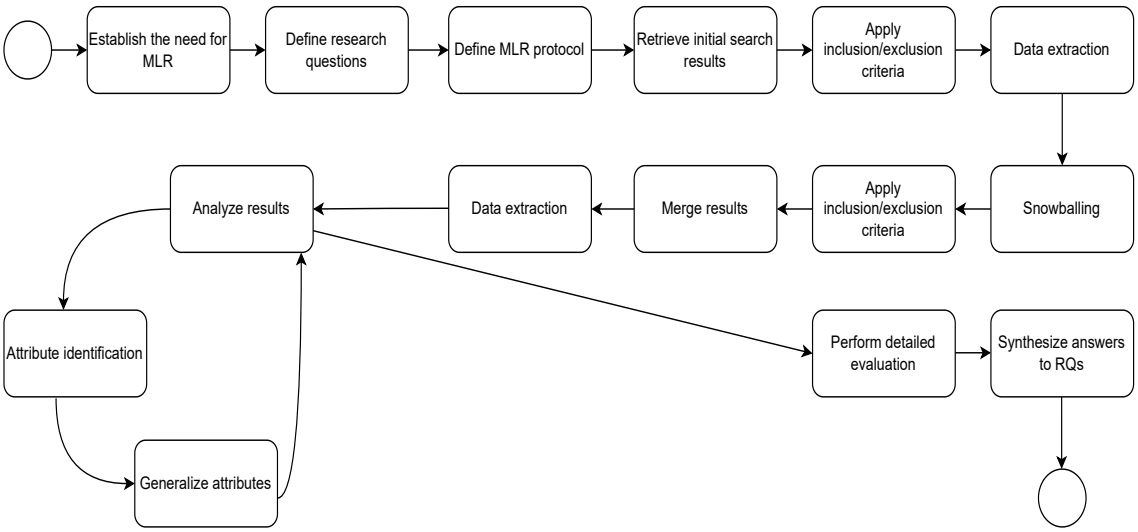

Figure 4.1: Overview of Multivocal Literature Review

this phase, research questions are raised that are answered by MLR and it is covered in section 4.1. Then we define and implement an MLR protocol. Both SLR and GLR follow the same protocol except for snowballing in SLR. The protocols helped us plan the study and systematically do the research. Then, we chose the search source for formal literature (IEEE Xplore, ACM Digital Library, Springer Link, ScienceDirect) and grey literature (Google and Bing). For both formal literature sources, and grey literature sources, search strings were formulated concerning their respective search methods. Depending on the stopping criteria defined for both SLR and GLR, initial search results were retrieved. The next step was applying corresponding inclusion and exclusion criteria for both SLR and GLR search results. Once we have the start set, data extraction was applied to extract data items needed to answer the RQs [32]. In the case of SLR, if we find the paper relevant and it has answers to our research questions, forward and backward snowballing was done to find some more relevant papers. Forward snowballing identifies new papers based on the citations from Google Scholar and backward snowballing uses a reference list in the paper. Filter criteria were applied to these new sets of papers from snowballing and it was merged with the initial set of SLR papers. The data extraction was done on this final set of SLR papers and GLR sources, and the results were analyzed. The data must be synthesized during this data analysis phase to properly respond to the RQs [32]. While analyzing the results, attributes are identified and generalized continuously as we go through all the sources. In our case, attributes are different challenges in the MLOps pipeline. Finally, a detailed evaluation is performed and answers to the research questions are derived.

4.2.3 Grey Literature Review Protocol

Selection of Sources

The two below web search engines are used as search source:

- **Google:** https://www.google.com/advanced_search
- **Bing:** https://www.bing.com

The selection of search engines is based on an informal pre-search utilizing the below keywords as well as the relevance of search results we have discovered in our prior experience. Instead of using sources like Stack Overflow, Stack Exchange, etc., we have discovered sites with data on MLOps that are more relevant from Google and Bing. As a result, we focused just on these specific search engines and websites.

Search Method

The same search method was applied for both Google and Bing.

Search functionality: Adding "AND", "OR" and quotes for phrases to the search strings helped us narrow the search; in our case, we used terms like "mlops pipeline" OR "Production ML“ which produced some better results than a standard search.

Set search location to the United States of America and use the following suffixes to exclude unwanted sites.

-site:youtube.com -site:researchgate.net -site:scholar.google.com -site:books.google.com -site:arxiv.org -site:springer.com -site:ieee.org -site:computer.org -site:acm.org -site:semanticscholar.org -site:amazon.com

Following four search strings were used in both search engines.

- "mlops challenges“
- "machine learning" ("continuous integration")
- "mlops pipeline" OR "Production ML“
- "mlops framework" OR "mlops practices"

The main idea was to use keywords that will help to answer the research questions. So we started with search strings such as "mlops challenges". Since MLOps involves DevOps part of software engineering, it was reasonable to include machine learning with continuous integration which led to the second search string "machine learning" ("continuous integration"). It gave insights into CI/CD practices in machine learning systems and their challenges and solutions. The last two search strings were used to understand the way MLOps are implemented in the industries and their best practices.

Stopping Criteria

The substantial amounts of data we received have an impact on the stopping rules. For example, when searching for "mlops pipeline" OR "Production ML“ we received 56500 hits from Google, and 36300 results from Bing. Obviously, in such cases, we have to limit our search results. The first 100 hits were considered. The search didn't need to go on further because the last pages didn't give any more useful search results.

- Google: first 100 URLs per search term, i.e., 100*4 (search strings) = 400 URLs
- Bing: first 100 URLs per search term, i.e., 100*4 (search strings) = 400 URLs

That means we have totally viewed 800 URLs. These URLs were exported to Microsoft Excel using Linkclump, which is a browser extension for link grabbing. All the duplicate URLs/rows were removed by clicking remove duplicates in the data tab of Microsoft excel. Once we have the initial set after de-duplication, inclusion and exclusion criteria mentioned below were applied in the first round to eliminate some unrelated URLs. The selected URLs from the first round went through the second set of criteria related to the quality assessment given below. In this round, some more irrelevant URLs were removed and we finalized 69 URLs from GLR for our study. Data extraction, MLOps challenges and solutions identification, and synthesizing answer for research questions was done as described at the end of this chapter.

Inclusion/Exclusion Criteria

We have created the following inclusion and exclusion criteria based on the goals of our research.

- Online articles or presentations of practitioners are desired
- Allowed formats:
  - Blog posts

  - Tutorials (in written form)
  - Company white papers (PDFs)
  - Presentation slides
  - News
- Exclude if:
  - Source is not in English
  - Scientific literature
  - Books
  - Videos
  - Announcement or description of a conference, seminar, training, etc
  - Job offerings

Quality Assessment (Relevance and Quality Filtering) The process of obtaining data from the search results included a quality assessment. We would conduct the quality assessment for all 260 sources. We would reject or exclude the results after doing the quality evaluation, therefore, we did remove 190 search results from our analysis based on the relevance and quality filtering that followed.

Relevance filtering: The goal is to find the materials that directly address a research query. We have developed the following relevance criteria concerning the goals of our research:

1. What are the challenges faced in different components of the MLOps pipeline by practitioners in the industry
2. What are the solutions to those challenges
3. What are the industry best practices for building MLOps pipeline

The search result should answer any of the research questions. If a search result meets at least one of the requirements listed above, it will be used for the study.

Quality filtering: At least one of the following conditions should be met to consider the result as the qualitative source of data for our study.

1. The article should be from a reputable organization

2. The author should have experience in the area or he should have published articles in this field
3. The article should not just be a hypothesis

4.2.4 Systematic Literature Review Protocol

Selection of Sources for SLR

The below sources are used for systematic literature review:

- **IEEE Xplore:** https://ieeexplore.ieee.org
- **ACM Digital Library:** https://dl.acm.org
- **Springer Link:** https://link.springer.com
- **ScienceDirect (Elsevier):** https://www.sciencedirect.com
- **Google Scholar (only for forward snowballing):** https://scholar.google.com

IEEE and ACM Digital Library give access to bibliographic literature of original research and development in this area. Springer and Science Direct help to find the related journal articles, and chapters from ebooks. The last source Google Scholar is used for tracking down citations of a particular document.

Search Method The same search method was applied for all four sources mentioned above.

Search functionality: It includes using boolean operators like "OR" and searching is done based on title and metadata to produce better results.

Following search string was used.

- Title: "mlops"
- Metadata: ("continuous integration" "machine learning") OR ("devops" "machine learning") OR ("dataops" "machine learning")

The first search was finding all the papers having MLOps in their title. Since there are very few papers related to MLOps we didn't consider specific search strings like "mlops challenges". In the second search string, the machine learning keyword was combined with continuous integration and DevOps to understand how CI/CD is implemented in a machine

learning system. DataOps is also used to understand how the data operations are done as it gives insights into building a data pipeline.

Inclusion and Exclusion Process A paper had to pass a two-step process to be included.

Basic Inclusion Criteria:

- Language should be English
- Publishing date from 2010 to 2022
- Only the first 200 results as per relevance.

The basic inclusion criteria was specified in advanced search box of all the sources. The results from all the sources was extracted to Mendeley. All the duplicate results were removed. The below two criteria were applied.

Content-Based Inclusion: Once the paper passes basic inclusion criteria it has to pass below two inclusion criteria.

1. Title-based inclusion
   - The search terms, or a combination of search terms, should appear in the title.
   - When this wasn't the case, we looked into alternative ideas, such as scanning for keywords in the paper to see if it contains any search terms.
   - A paper was immediately disqualified from the research if it could be inferred from its title that it was inappropriate. If it passes this stage or the title could not be evaluated it still had to pass the abstract-based inclusion.
2. Abstract-based inclusion
   - Are the authors specifically addressing MLOps challenges, solutions, continuous deployment in ML applications, or related terms (e.g. continuous delivery, continuous experimentation and monitoring in ML)? We excluded a paper if this was not the case.
   - Is there a sufficient focus on building MLOps pipeline and their practices? Other sections of the article, such as the conclusion, were skimmed if the answers to these questions could not be found by reading the abstract alone.

After the initial search, and after applying two rounds of filter criteria and snowballing, we had 29 papers from SLR.

Snowballing Data extraction was done on SLR papers as described below. All the papers from SLR went through snowballing process to find some more relevant papers. Snowballing is a technique that is adapted to the research methodology for all kinds of literature reviews in order to find more new publications related to the research. There are two types of snowballing, forward and backward snowballing. Forward snowballing identifies new papers based on those papers citing the paper being examined [34]. Google Scholar is used for this and the new set of papers is identified. The backward snowballing uses the reference list of the inspected paper to identify new papers that could be included [34]. The newly retrieved set of papers is now evaluated as described above. i.e., it goes through the inclusion and exclusion process. No additional search for this paper was conducted.

Data extraction to Synthesize Answers for RQs The data extraction was done on the first set of papers from the initial search of SLR, the second set of papers from snowballing, and the final set of URLs from GLR. The data extraction technique was similar in both SLR and GLR. Papers from SLR and URLs from GLR were studied to identify the MLOps challenge and solutions. Two separate excel sheets were maintained to document challenges mentioned in SLR and GLR. As we went through the sources, we collected different MLOps challenges, solutions, and tools mentioned in them. We documented what challenges and solutions are mentioned by each source so that we have a final overview of what challenges appear more frequently. If a new challenge appears in a paper or URL the first attempt was to check if it belongs to any existing MLOps challenge, if it does not it was considered a new challenge otherwise we would add to the existing challenge and generalize it. The same process went on for all the sources and we had a final list of MLOps challenges and solutions from SLR and GLR in separate excel sheets. This answered our first two research questions. For the third research question (RQ3), the research guidelines were synthesized from RQ1 and RQ2. Depending on what guidelines could be practically implemented with the available resource, only those guidelines are chosen for implementation (RQ4).

5 Results

In this chapter, the results of MLR are shown in the first section. Section 5.2 presents the answers for RQ1 and RQ2 from multivocal literature review. Section 5.3 discusses the third research question (RQ3).

5.1 Search Results of Multivocal Literature Review

The number of search results we included or excluded for all search sources for each phase of the filtering criteria is shown in figure 5.1 and 5.2 for SLR and GLR respectively.

As shown in table 5.1, there are 29 final results from SLR. We gathered most of our research result from IEEE Xplore (14), ACM Digital library (07) followed by Science Direct (05) and Springer Link (02).

Table 5.1: Results in all filtering stages of SLR

Data Source	**Basic IC**	**Title IC**	**Abstract IC**	**Data extraction**	**Snowball(all filter criteria and data extraction applied)**	**Total(Data extraction Column + Snowball Column)**
IEEE Xplore	101	53	17	08	06	14
ACM Digital Library	200	30	17	05	02	07
Springer Link	15	15	14	01	01	02
ScienceDirect	12	12	06	03	02	05
Total						29

In case of GLR, there are 69 total results as shown in table 5.2. There are three filtering stages and during these three phases we selected more Google grey literature sources (49) than results from Bing (30).

Table 5.2: Results in all filtering stages of GLR

Data Source	**Stopping criteria/Deduplication**	**Inclusion/Exclusion criteria**	**Quality Assessment**
Google	345	150	49
Bing	290	110	30
Total			69

5.2 Challenges (RQ1) and Solutions (RQ2) within MLOps

The answers to two research questions (RQ1 and RQ2) are extracted from 29 papers from SLR and 69 URLs from GLR as mentioned in Appendix A.2 The data extracted from these sources are merged and presented in table 5.3. There are 19 challenges from SLR and 20 challenges from GLR that were reported in our literature review. Most of the challenges appeared in both SLR and GLR except for a few which were specific to SLR such as *Coupling between system and model requirements*, and *Component entanglement*. The other two challenges that were specific to GLR were *Incompatibilities between development and production, Security and privacy*, and *Forgetting the downstream application of a new model*. The solutions and tools mentioned to these challenges from all the sources are aggregated and categorized in the table. These solutions help us give an insight on how these problems for MLOps challenges are overcome in the industry and academia. There are some challenges for which no solutions exist such as *Operations and feedback loops*, *Explainability*, *Coupling between system and model requirements*, *Component entanglement*, *Multitude of tools*. Finally, the MLOps challenges are categorized into 4 groups namely Data Management Challenges, Model Creation Challenges, Model Deployment Challenges, and Cross Cutting Concerns. Table 5.3 presents the category of challenges, name of challenge , solution, and tools from our literature review in brief. Table A.1 in Appendix presents it in detail.

Table 5.3: Brief Description of MLOps challenges and Solutions from Literature Review

Category	**Challenge**	**Solution**
Data Management	Data access and management	Datalake
Data Management	Shortage of diverse data samples	Resampling, Augmentation
Data Management	Data cleaning and validation	Data scrubbing
Data Management	Data labeling	Use trained model to label
Model Creation	Feature selection	Feature store
Model Creation	Calculation of performance metrics	River
Model Creation	Algorithm and hyperparameter selection	Automl
Model Creation	Model evaluation	Deepchecks
Model Creation	Experiment tracking	DVC
Model Deployment	Model monitoring	Model picker, Evidently AI
Model Deployment	Managing deployment pipelines	Efficient automated pipelines right from the development
Model Deployment	Operations and feedback loops	
Model Deployment	Incompatibilities between development and production	Staging and canary integration tests
Cross-Cutting Concerns	Vendor lockin with commercial Mlops platform	Devise a MLOps platform with open source frameworks
Cross-Cutting Concerns	Difficulty to estimate infrastructure resources	Automatic scaling
Cross-Cutting Concerns	Explainability	
Cross-Cutting Concerns	Security and privacy	Security and access control
Cross-Cutting Concerns	Forgetting the downstream application of a new model	create a working skeleton initially
Cross-Cutting Concerns	Coupling between system and model requirements	

Continued on next page

Table 5.3: Brief Description of MLOps challenges and Solutions from Literature Review (Continued)

Category	Challenge	Solution
Cross-Cutting Concerns	Component entanglement	
Cross-Cutting Concerns	Multitude of tools	

Literature Review Analysis Figure 5.1 gives an overview of the literature review analysis of all the challenges. Model monitoring, Experiment tracking, Data cleaning and validation are the top 3 challenges discussed in both SLR and GLR.

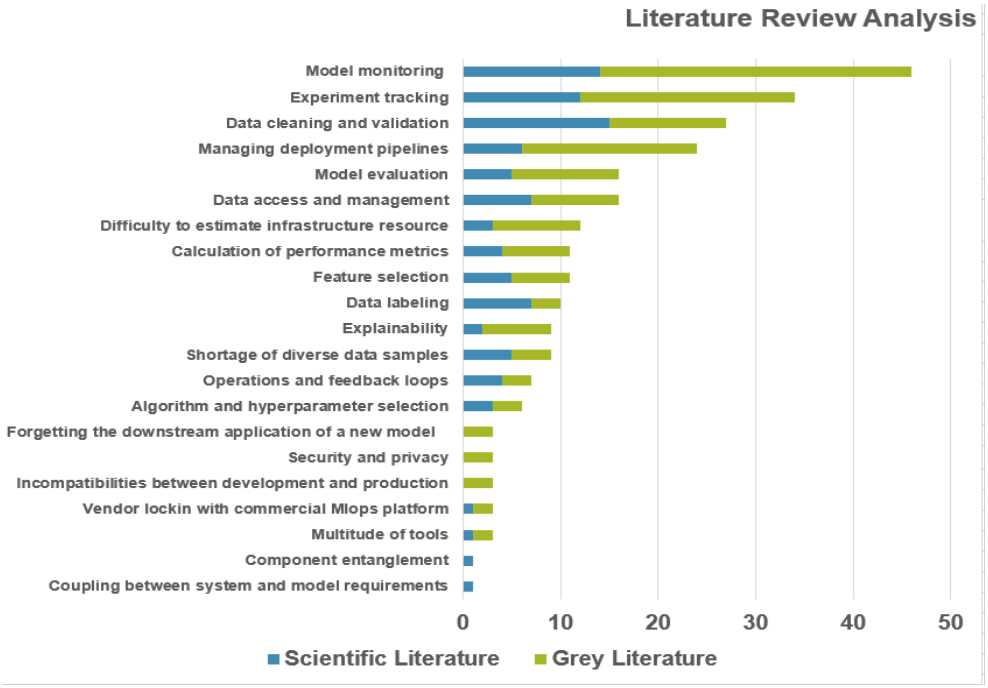

Figure 5.1: Literature Review Analysis

The category of challenge that is most prominent in SLR and GLR are model deployment challenge followed by model creation and data management. These categories of challenges are discussed in detail below.

Data Management Challenges All the challenges that occurred in the data manipulation pipeline such as data access and management, shortage of diverse data samples, data cleaning

and validation, and data labeling were grouped. Figure 5.2 shows the number of SLR and GLR resources that reports these challenges. Data cleaning and validation is the most discussed challenge.

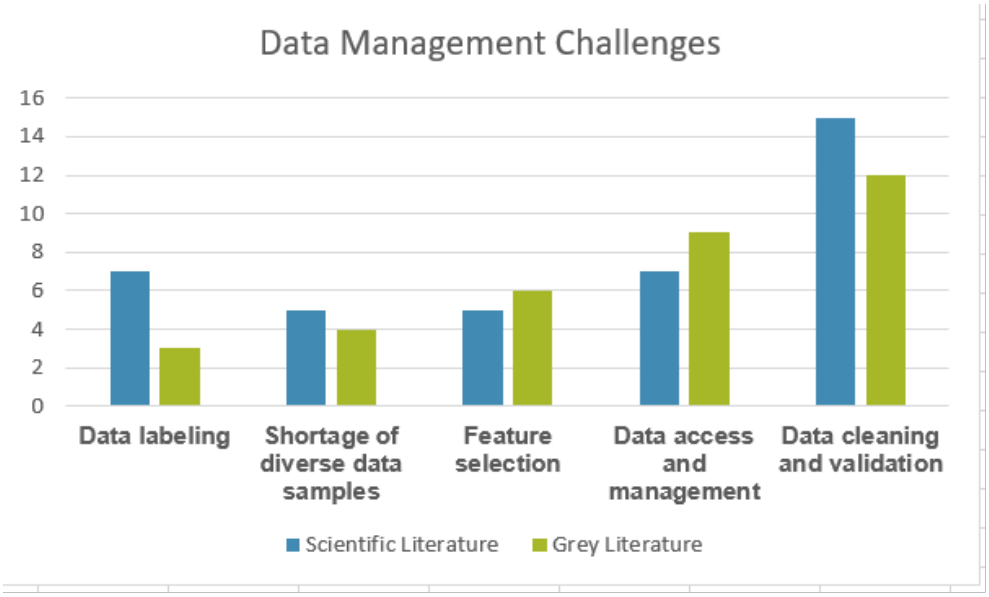

Figure 5.2: Data Management Challenges

Model Creation Challenges There are a number of challenges that are encountered during building and evaluating models, namely feature selection, calculation of performance metrics, algorithm and hyperparameter selection, model evaluation, experiment tracking. The distribution of these challenges is presented in figure 5.3. experiment tracking is the most frequently encountered challenge followed by model evaluation.

Model Deployment Challenges The problems faced while deploying ML models into production are discussed in this category. It includes operation and feedback loops, incompatibilities in production and machine learning, managing deployment pipelines, and model monitoring being the most popular challenge as shown in figure 5.4.

Cross Cutting Concerns These challenges include the ones that do not fall into the above three categories such as Vendor lock-in with the commercial MLOps platform, difficulty to estimate infrastructure resources, explainability, security and privacy, forgetting the downstream application of a new model, coupling between system and model requirements, component entanglement, and multitude of tools as shown in figure 5.5.

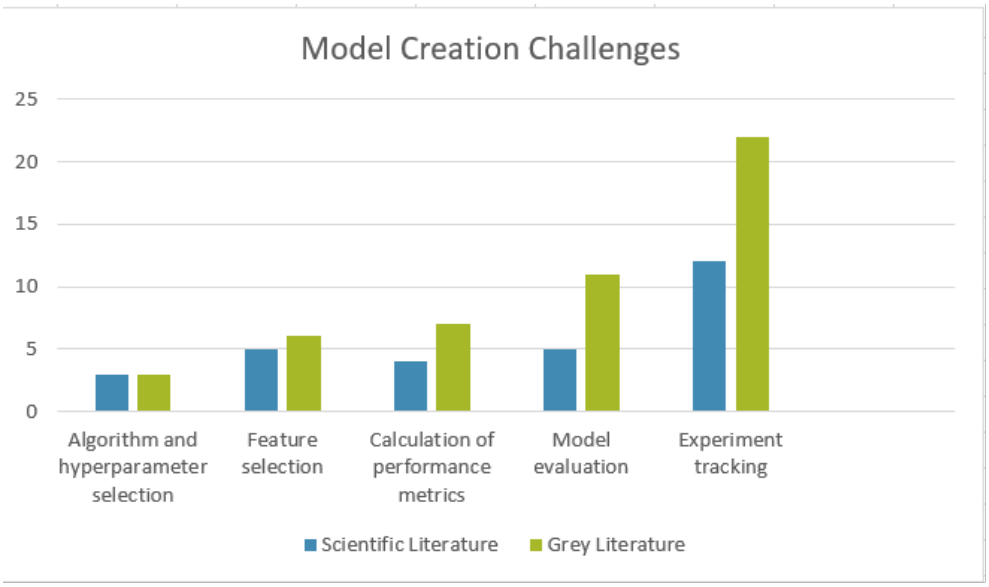

Figure 5.3: Model Creation Challenges

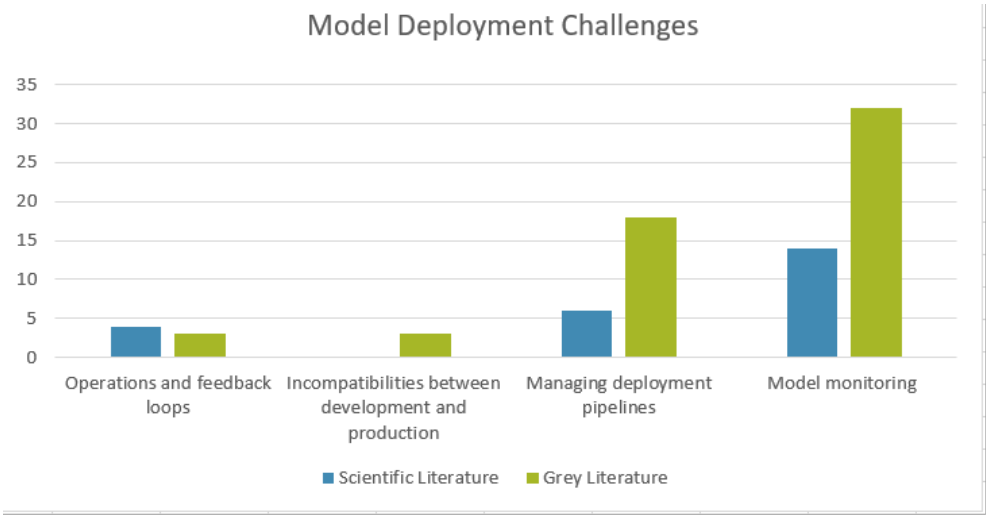

Figure 5.4: Model Deployment Challenges

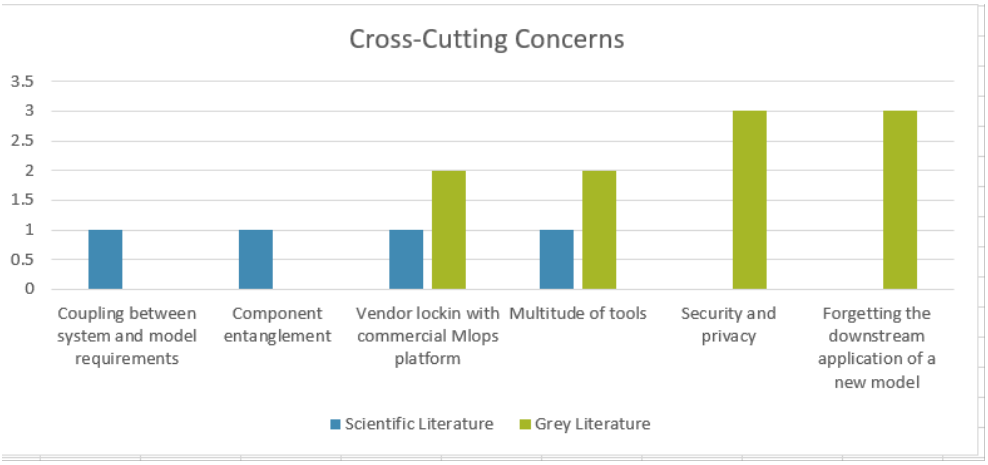

Figure 5.5: Cross Cutting Concerns

5.3 MLOps Guidelines for Practitioners (RQ3)

Based on the results from RQ1 and RQ2, the answers for RQ3 are derived. Here, seven practical guidelines are proposed to practitioners, enriched with the available tool support. The challenges that are related to each other are grouped together and a single guideline is proposed for them. For example, data access and management, and shortage of diverse data samples challenges can be solved by setting up a data lake that not only accesses data from various sources but also considers a variety of data. Some challenges that could not be combined are considered individually, and a guideline is proposed. Only those challenges that have enough tool support are considered for synthesizing guidelines. Seven guidelines are proposed and they are presented in the table 5.4. below.

Table 5.4: MLOps Guidelines for Practitioners based on RQ1 and RQ2

	Guideline
1	Set up a data lake with a diverse set of data samples
2	Scale feature engineering in your organization with a feature store
3	Build a simple model initially and ensure its infrastructure compatibility
4	Perform thorough analysis and cleaning of the training data
5	Choose cost effective data labelling techniques

Continued on next page

Table 5.4: MLOps Guidelines for Practitioners based on RQ1 and RQ2 (Continued)

	Guideline
6	Use an efficient monitoring system
7	Use experiment tracking to achieve reproducibility

The guidelines are explained in detail below.

1. Set up a data lake with a diverse set of data samples

Challenges Addressed: Data access and management, Shortage of diverse data samples

Motivation: Data comes from various sources and it can be in various forms like structured, unstructured, and semi-structured. Data can be static, batch, streaming, and real-time. The absence of a data lake leads to the creation of the same data pipeline to the same source multiple times. The lack of diverse data samples leads to the poor performance of the model.

Description: Data is collected from different sources and stored in the data lake as shown in figure 5.6. Data lakes are systems or repositories where data is kept in its original, unprocessed form, typically as objects or blob files. It handles data throughout its lifecycle and analyzes data in a variety of ways. This data can be transformed and queried directly to get the required information with the help of an external API.

Benefits: It can store data in its native format and process any variety of it, ignoring size limits. The data is now easy to access for most people within an organization

Tools: Data lake on AWS [1] , Azure data lake storage [2]

2. Scale feature engineering in your organization with a feature store

Challenges Addressed: Feature selection

Motivation: Features are important for models to predict the output. Features are not directly sent from real-world systems instead they send raw data. Features should be extracted from the raw data. More time is spent on creating features and managing these fragmented data infrastructures. For every model, the feature extraction is done from scratch which leads to duplication of work. Scaling this job when data grows is also very hard.

[1] https://aws.amazon.com/solutions/implementations/data-lake-solution/
[2] https://azure.microsoft.com/en-us/services/storage/data-lake-storage/

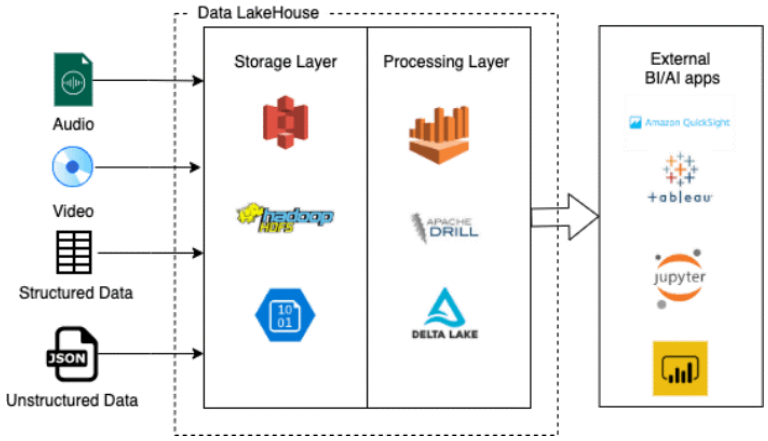

Figure 5.6: Datalake[3]

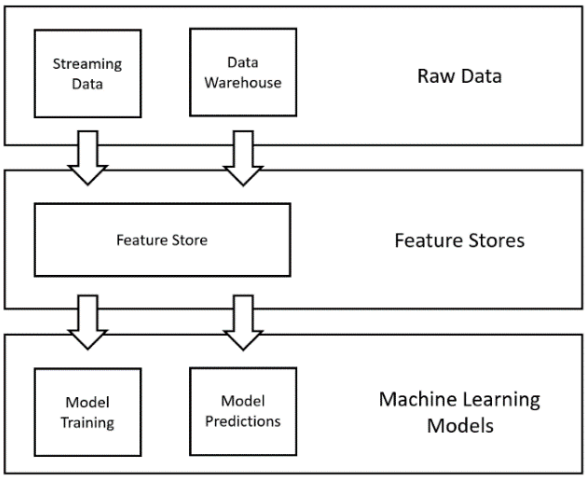

Figure 5.7: Feature store[4]

Description: Feature store is a centralized repository for feature management and discovery as shown in figure 5.7. All the teams can use the same feature if required without duplication. It supports both stream and batch data, which goes into online and offline feature stores respectively. Data is consistent in both the stores and it is used for model training and serving.

Benefits: Features can be shared between teams therefore the time taken to develop models decreases. Training serving skew can be avoided by having a monitoring capability to the feature store so the developer is notified when there is a change in the features in produc-

[3]https://medium.com/adfolks/data-lakehouse-paradigm-of-decade-caa286f5b7a1
[4]https://towardsdatascience.com/design-patterns-in-machine-learning-for-mlops-a3f63f745ce4

tion.

Tools: Feast [5], Tecton [6]

3. Build a simple model initially and ensure its infrastructure compatibility

Challenges Addressed: Managing deployment pipelines, Incompatibilities between development and production

Motivation: It takes a lot of time and labor to manually retrain and validate models because of their iterative nature. Model building is a part of it, the actual product is the whole pipeline. Data scientists usually concentrate on building complex models with high accuracy. Having a complex model for the first deployment introduces a lot of complications.

Description: Building a simple model with an automated CI/CD pipeline and checking for its infrastructure compatibility helps in easy deployment procedures as shown in figure 5.8. It is important to do certain checks such as whether the data reaches the data lake properly from various sources, whether data is getting converted to features and reaching the model, is the model being trained with correct parameters, is model responding to prediction requests without latency, etc. Once there is strong compatibility between different stages and CI/CD is working fine in the development environment, data scientists and ML engineers can think of improving the model and deploying it to production.

Benefits: Reduces time and effort as we don't end up with unnecessary complications because we start with a simple model. Building and testing a more complex model becomes easy. It avoids incompatibility in development and production environments.

Tools: MLflow [7], Kubeflow [8], TensorflowExtended [9], DVC [10]

4. Perform thorough analysis and cleaning of the training data

Challenges Addressed: Data cleaning and validation

Motivation: Dirty data and missing data affect model performance directly. Data exploration is about understanding the data to know what each column describes and detect anomalies in the data. Data cleaning is an important step and takes up a significant amount of time.

[5] https://feast.dev/
[6] https://www.tecton.ai/
[7] https://mlflow.org/
[8] https://www.kubeflow.org/
[9] https://www.tensorflow.org/
[10] https://dvc.org/

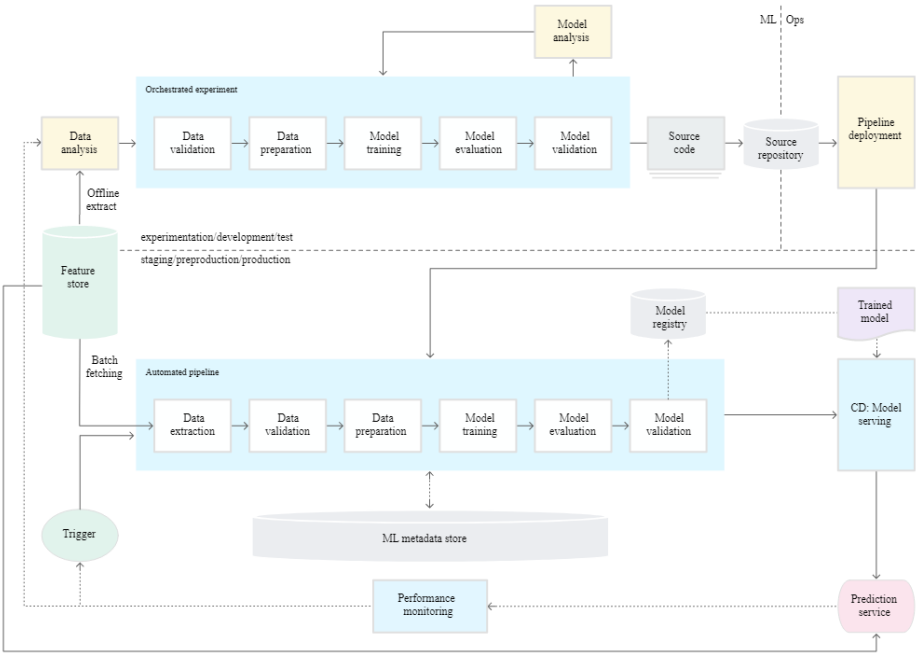

Figure 5.8: MLOps pipeline[11]

Description: Define a schema for the data during training and follow the same schema for inference as well. Modify or remove incomplete, incorrect, inaccurately formatted, or duplicate data in a dataset to improve consistency, accuracy, and reliability. Normalization/standardization techniques are commonly applied to numerical data before training an ML model to deal with outliers and make the data look more like a Gaussian distribution.

Benefits: Clean data increases the accuracy of the model by reducing the error margin.

Tools: CPClean [12]

5. Choose cost effective data labelling techniques

Challenges Addressed: Data Labelling

Motivation: Labeled datasets support the development of machine learning models that recognize and comprehend recurrent patterns in the input to produce accurate output. If the data is not labeled, it is important to find the labels of the data quickly and cost-effectively.

[11] https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

[12] https://ease.ml/components/cpclean/

Description: Data labeling can be done efficiently by taking advantage of ML models and the knowledge of domain experts as shown in figure 5.10. An ML model can be trained from human-labeled data initially. This model can be used later to predict the labels for unlabeled data. If there is any ambiguity, then the data is sent to domain experts.

Benefits: It takes less time to label data, as multiple groups are involved.

Tools: Amazon sage maker ground truth [13], Labelbox [14]

6. Use an efficient monitoring system

Challenges Addressed: Calculation of performance metrics, Operations and feedback loops, Model Monitoring

Motivation: Models often become stale over time and start underperforming. Model metrics depend on the ground truth/labels. When the value of these metrics deteriorates in production, there is a divergence between training performance and prediction performance in production, which should be detected via monitoring.

Description: Monitoring systems should have three components as shown in figure 5.9. First, Processing and storage to store three kinds of metrics related to data monitoring, prediction monitoring, and system monitoring. The second component is Visualization to pick up trends and changes in the data. The third component is Alerting to notify the engineers when there is training-serving-skew.

Figure 5.9: Monitoring system

Benefits: It ensures that a well-performing model stays in production among all the available models.

Tools: Amazon Sage maker model monitor [15], Evidently AI [16], Fiddler [17]

7. Use experiment tracking to achieve reproducibility

Challenges Addressed: Algorithm and hyperparameter selection, Experiment tracking

[13] https://aws.amazon.com/sagemaker/data-labeling/
[14] https://labelbox.com/
[15] https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html
[16] https://evidentlyai.com/
[17] https://www.fiddler.ai/

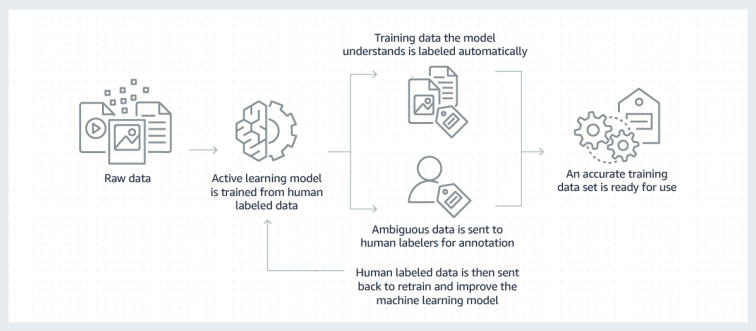

Figure 5.10: Data labelling[18]

Motivation: Building ML models does not involve only code, but also metrics, parameters, and data. It is difficult to select the right algorithm and parameters for a problem. Therefore, a series of experiments are performed until the model achieves sufficient accuracy.

Description: Models are evaluated with various parameters and algorithms. All these results are tracked by versioning data, parameters, and metrics used for each experiment. It is important to keep track of all these for provenance and for reverting to a previous model version.

Benefits: It helps to test multiple different approaches. Old models can be quickly rolled back to production.

Tools: DVC [19], MLflow [20]

[18] https://aws.amazon.com/sagemaker/data-labeling/what-is-data-labeling/
[19] https://dvc.org/
[20] https://mlflow.org/

6 Practical Demonstration of Guidelines

This chapter answers fourth research question (RQ4) and presents the details of practical implementation of guidelines discussed in the previous chapter.

6.1 Guidelines Chosen for Practical Demonstration (RQ4)

Guidelines are demonstrated on our local setup. A local development environment (LDE) is a means to set up services on our laptop or desktop to run a website or a web application. Few of the guidelines are difficult to demonstrate because we do not have access to real-world data from various sources(batch or streaming data). Since we are taking toy datasets from Kaggle we do not come across the problem of missing labels, cleaning data, or creating the need for a feature store as we do not have a large amount of data and multiple teams working on this data. Therefore, we chose the below three guidelines for practical demonstration as its feasible to demonstrate them with all the available sources in our local environment.

1. Build a simple model initially and ensure its infrastructure compatibility.
2. Use an efficient monitoring system.
3. Use experiment tracking to achieve reproducibility.

To demonstrate the above guidelines, open source tools are used and the details are presented below.

6.2 Dataset and Tools

Dataset

Dataset used for our practical demo is Diabetes Dataset of Pima Indian Heritage. The National Institute of Diabetes and Digestive and Kidney Diseases is the original source of

this dataset. The goal is to determine whether a patient has diabetes based on diagnostic parameters [20].

Number of Instances: 768
Number of Attributes: 8 plus class
For Each Attribute: (all numeric-valued)

1 Number of times pregnant

2 Plasma glucose concentration after 2 hours in an oral glucose tolerance test

3 Diastolic blood pressure (mm Hg)

4 Triceps skin fold thickness (mm)

5 2-Hour serum insulin (mu U/ml)

6 Body mass index (weight in kg/(height in m^2))

7 Diabetes pedigree function

8 Age (years)

9 Class variable (0 or 1)

Missing Attribute Values: Yes
Class Distribution: (class value 1 is interpreted as "tested positive for diabetes")

Tools

Data Version Control (DVC) is an open-source version control system for machine learning projects [9]. It maintains version control for intermediate files, data sets, and machine learning models. DVC employs code to link them together and stores file contents on network-attached storage, a disc, Amazon S3, Microsoft Azure Blob Storage, Google Drive, and Google Cloud Storage. Every ML model's entire evolution may be tracked due to comprehensive code and data provenance. This provides for reproducibility and makes switching between experiments simple [9]. DVC pipelines are built when a certain set of actions update the project frequently such as gathering data, extracting features, model training, and evaluation. These stages are defined in `dvc.yaml` file [10].

Git is a DevOps tool for managing source code. It is a version control system that can efficiently manage small to very large projects and is free and open-source. Git tracks source

code changes, allowing several developers to collaborate on non-linear development [16]. It helps in continuous integration and continuous delivery.

Heroku is a cloud Platform as a Service that uses containers (PaaS). Modern apps are deployed, managed, and scaled by developers using Heroku. This platform provides developers with a quick route to releasing their apps onto the market since it is elegant, flexible, and simple to use [17].

Evidently AI is an open-source Python library designed for machine learning and data scientists. It aids in assessing, testing, and keeping track of ML model performance from validation to production [11]. It provides interactive visual HTML reports from pandas.DataFrame or csv files used for documentation, model evaluation.

Visual studio code is the IDE (Integrated Development Environment) used for source code development. It is a simplified code editor that supports development activities like task execution, debugging, and version management [33]. Python [1] is our coding language and libraries like sklearn [2], pandas [3], Flask [4] are used.

6.3 Build a Simple Model Initially

The main aim of this guideline is to build a simple model with an automated CI/CD pipeline and check for its infrastructure compatibility which helps in easy deployment procedures. Initially, the basic functionalities like processing and splitting data, training model, and prediction requests are built and tested to ensure its working properly. Then we can add advanced features and try better algorithms for the training model so this way it reduces unnecessary complications. DVC is used to create pipelines and a GitHub Actions workflow is used to achieve CI/CD and deploy it on Heroku.

To demonstrate this guideline, firstly DVC needs to be installed using pip (Package installer for python). A DVC pipeline is built which consists of process_data, split_data, train_and_evaluate as shown in listings 6.1 in a dvc.yaml file defined in our project. Every stage in dvc.yaml file has its name, dependencies (deps) such as input file and dataset file, and output file (outs). The output of one stage is the input to the next stage. This is

[1] https://www.python.org/
[2] https://scikit-learn.org/stable/
[3] https://pandas.pydata.org/
[4] https://flask.palletsprojects.com/en/2.2.x/

depicted well by the DVC data acyclic graph shown in figure 6.1. This graph appears when `dvc dag` command is typed in terminal.

In the process_data step, a raw dataset is considered. If the names of columns contain space it is replaced by an underscore and saved in the processed folder. This processed data is sent to the next step split_data, and data is split into train and test data and stored in their respective folders. The next step is train_and_evaluate where the model is trained and evaluated. The diabetes dataset has labels so we have considered a supervised learning classifier, the k-nearest neighbors algorithm, sometimes referred to as KNN or k-NN, that employs proximity to produce classifications or predictions because it relies on the idea that comparable points can be discovered close to one another [18]. The parameter for this algorithm is the number of neighbors. We have defined four metrics such as accuracy, precision, recall, and f1-score to assess this model.

These changes are committed to the Git repository. The used GitHub Actions workflow is defined in `cicd_for_ml.yaml` file as shown in listing 6.2. It names the branch in which the changes should be deployed, specifies the dependencies to be installed, executes the unit test cases, and if all these step passes, the app is deployed on Heroku. Before deploying on Heroku, the app must be registered on the platform and an api_token is generated for each registered app and it should be used while accessing it. This way continuous integration and continuous delivery are achieved by automatically deploying the app every time the changes are pushed to GitHub.

Listing 6.1: dvc.yaml

```
stages:
  process_data:
    cmd: python src/process_data.py --config=params.yaml
    deps:
    - src/process_data.py
    - realworld_data/raw/Diabetes.csv
    outs:
    - realworld_data/processed/Diabetes.csv

  split_data:
    cmd: python src/traintest_split.py --config=params.yaml
    deps:
    - src/traintest_split.py
    - realworld_data/processed/Diabetes.csv
```

```
    outs:
    - feature_data/train/Diabetes_train.csv
    - feature_data/test/Diabetes_test.csv

  train_and_evaluate:
    cmd: python src/train_model.py --config=params.yaml
    deps:
    - feature_data/train/Diabetes_train.csv
    - feature_data/test/Diabetes_test.csv
    - src/train_model.py
    params:
    - estimators.KNN_Nearest_Neighbor.params.neighbors
```

Figure 6.1: Result of dvc dag

Listing 6.2: cicd_for_ml.yaml

```
name: cicd_for_ml
on:
  push:
    branches:
    - main
  pull_request:
    branches:
    - main
```

```
jobs:
  build:
    runs-on: windows-latest
    steps:
    - uses: actions/checkout@v2
      with:
        fetch-depth: 0
    - name: Set up Python 3.10
      uses: actions/setup-python@v2
      with:
        python-version: 3.10
    - name: Install required dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt;
    - name: Test the unit test cases with pytest
      run: |
        pytest -v
    - name: Deploy to Heroku
      env:
        HEROKU_API_TOKEN: ${{ secrets.HEROKU_API_TOKEN }}
        HEROKU_APP_NAME: ${{ secrets.HEROKU_APP_NAME }}
      if: github.ref == 'refs/heads/main' && job.status == '
          success'
      run: |
        git remote add heroku1 https://heroku:615a0b93-b89d-44
            ea-a873-1b8ad1d505d7@git.heroku.com/diabetes-
            prediction-mlops.git
        git push heroku1 HEAD:main -f
```

The prediction app webpage looks as shown in the figure 6.2. The input should be provided for all the attributes depending on the range specified. This range is decided by considering min and max of each attribute in the whole dataset. The output is displayed as "diabetic" or "healthy" on the right side.

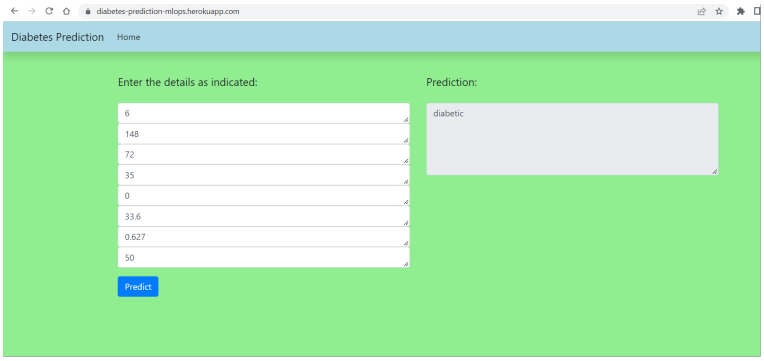

Figure 6.2: GUI of Diabetes prediction app

6.4 Use Experiment Tracking to Achieve Reproducibility

Models are evaluated with various parameters and algorithms. It is important to keep track of all these for reverting to a previous model version if there is unexpected bugs in a newly deployed model version or if the need arises. Therefore, experiment tracking is very important and it is implemented by git for code versioning with Git, DVC for data versioning, and MLflow for model tracking.

Code is versioned with Git as explained earlier. The DVC pipeline defined in the previous section versions the data and the whole pipeline. The external storage for DVC versioned data in our project is Google Drive as shown in figure 6.3. It stores the versioned data. Versioned folders are encoded to some numbers by DVC.

Model tracking is done using MLflow. It is installed using pip and the configurations such as the artifacts_dir, experiment_name, run_name, registered_model_name, remote_server_uri are specified in params.yaml file as shown in listing 6.3.

Listing 6.3: params.yaml

```
mlflow_config:
  artifacts_dir: artifacts
  experiment_name: KNearest Neighbor for Diabetes
  run_name: KNN
  registered_model_name: KNNforDiabetes
  remote_server_uri: http://127.0.0.1:5000
```

The model is trained using various parameters. The details of each experiment like parameters and metrics are stored and displayed in MLflow to compare and choose the best model for

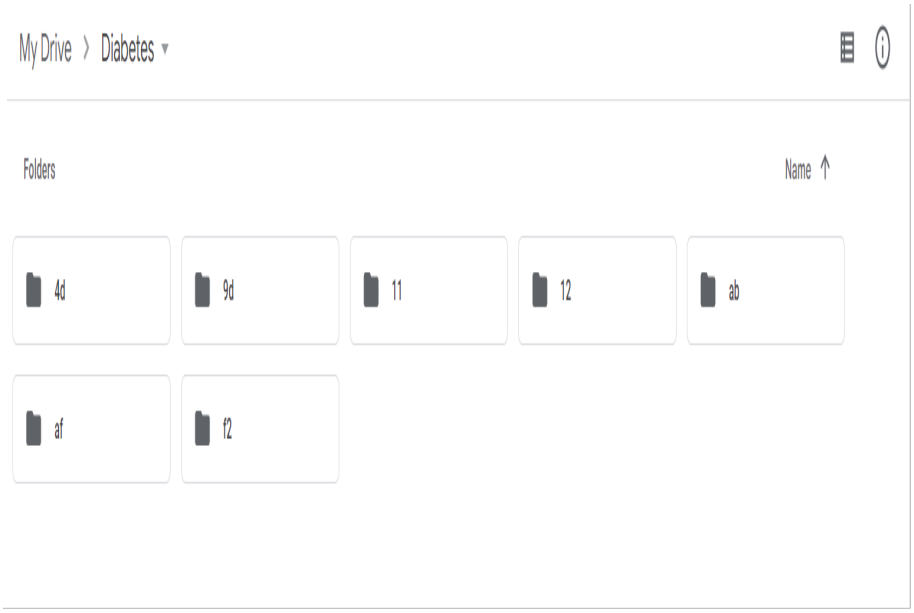

Figure 6.3: External storage for DVC

deploying in production. This is achieved by setting experiment and URI, and logging metrics and parameters using the MLflow command. In our experiment, the model is trained using the different number of neighbors like 1,3,5,7,9 and their results are recorded in MLflow as shown in figure 6.4.

MLflow has great visualization capabilities to compare all 5 runs from one experiment using various graphs like Parallel Coordinate Plots as shown in figure 6.5. One parallel axis specifies the number of neighbors and remaining four specifies metrics. It can be seen that accuracy is highest when the number of neighbors is 7 and above.

MLflow also has visualizations to compare all 4 metrics from all 5 runs together in a bar graph as shown in figure 6.6.

6.5 Use an Efficient Monitoring System

Model monitoring plays an important role to identify when the model becomes stale over time. When the value of these metrics deteriorates in production, there is a divergence between training performance and prediction performance. This is demonstrated by showing data drift monitoring using Evidently AI. Data drift is created by adding noise to the data and sending this data to a trained model to detect drifts in features.

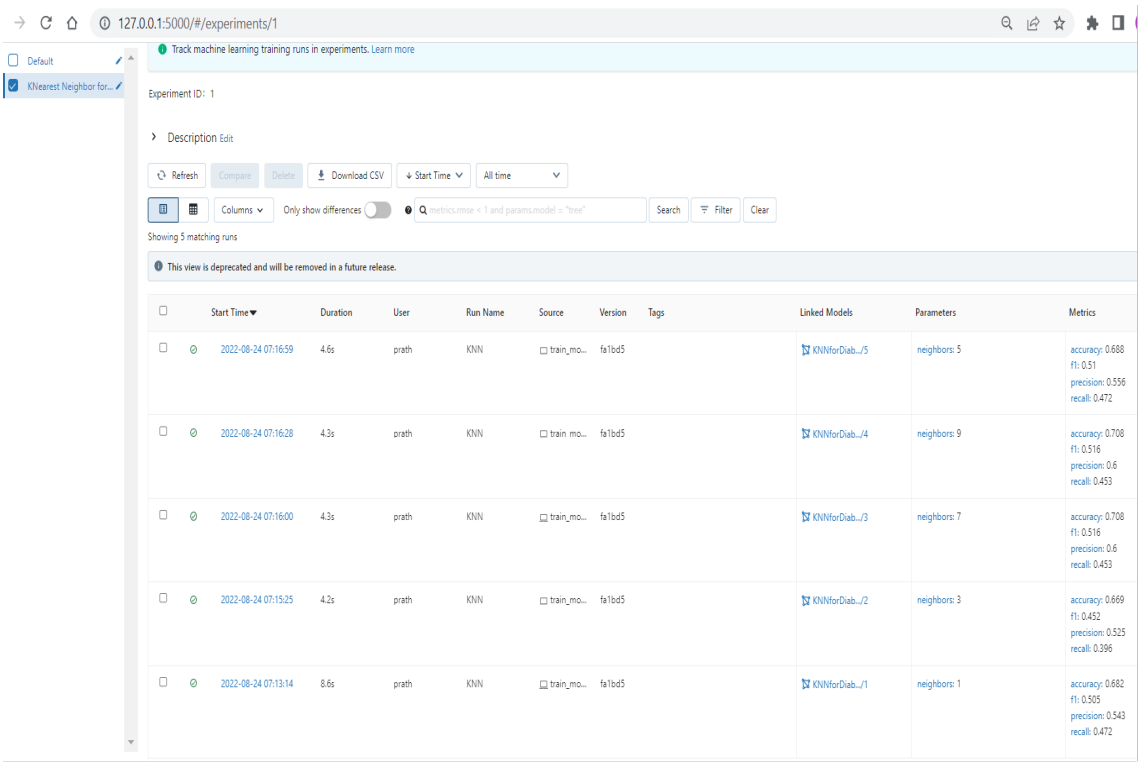

Figure 6.4: Experiment tracking in mlflow

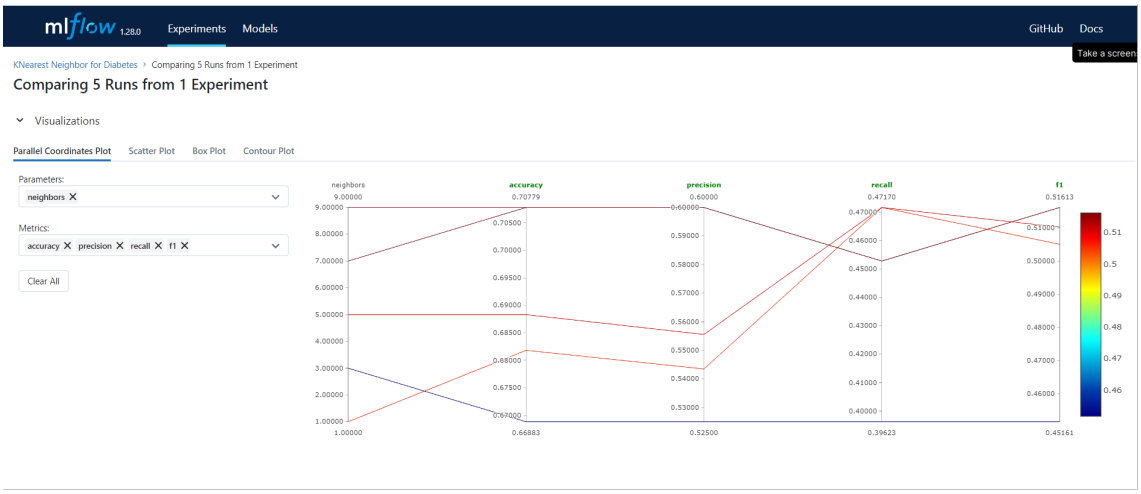

Figure 6.5: Comparing 5 Runs from 1 Experiment

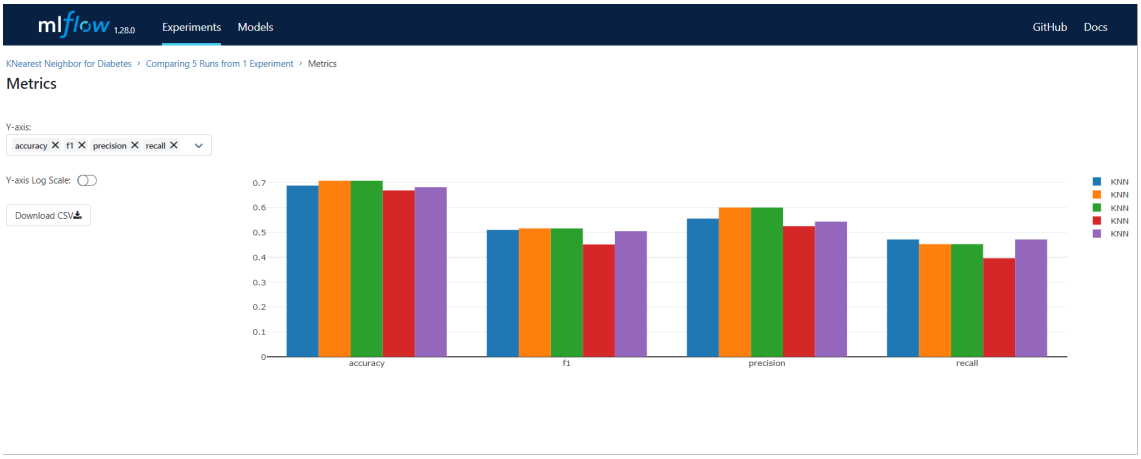

Figure 6.6: Metrics from all 5 runs

In a real-world case, model degradation happens when there is data drift or target drift. This is recorded by monitoring tools. In our case, we are not using real/streaming data to notice that drift. So, to demonstrate this guideline practically, Gaussian noise is added to the data. Gaussian noise is a probability distribution that depicts the random fluctuations in an ongoing physical process [13]. Evidently AI is used to monitor the drift in the noisy data and original data and changes in metrics are recorded for both models as shown in figure 6.7. It can be seen that the quality of the model is reduced and its accuracy is decreased to 0.662. and change in quality metrics is shown in figure 6.8.

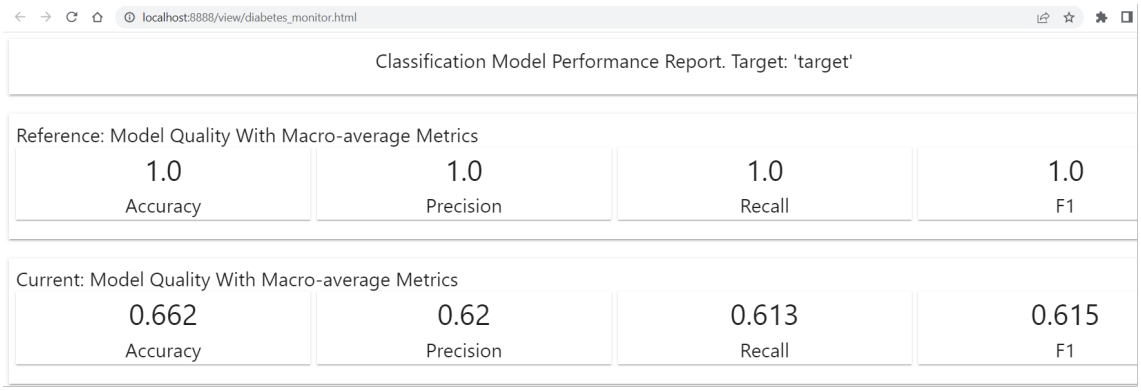

Figure 6.7: Model quality change for reference and current model

This tool also helps us to identify which attributes/features actually drifted. As shown in figure 6.9 the last 4 attributes namely Pregnancies, Skin_Thickness, Insulin, Diabetes_Pedigree_Function are drifted. But the drift score for the last two attributes is 0 because their data drift is

Figure 6.8: Quality Metrics by Class

very low which cannot be captured in 6 decimal digit numbers. Drift detection is 40% as mentioned at the top. Dataset drift is not detected because only 40% of features are drifted. Therefore there is only feature drift.

Figure 6.9: Dataset drift

7 Discussion

This chapter discusses the collected results and limitations.

This study highlights the challenges in various components of the MLOps pipeline and proposes practical guidelines based on it. Both SLR and GLR list the same set of MLOps challenges except for a few. The challenges which are only listed by SLR are *Coupling between system and model requirements*, *Component entanglement*. GLR also lists few unique challenges such as *Incompatibilities between development and production*, *Security and privacy*, *Forgetting the downstream application of a new model*. This is the case because GLR is a practitioner's experience where they take into account the real-world problems in developing ML applications like security and privacy of data, incompatibilities between various software environments, and also the application that is consuming the ML model. Some challenges appear more in GLR such as *Experiment tracking*, *Model monitoring*, *Managing deployment pipelines*. Regarding the solutions to these challenges, GLR presents a large number of tools available for each problem compared to SLR. SLR not only provides solutions from a tools perspective but also gives some other ideas like design patterns as solutions for MLOps challenges. For example, in *Data access and management* challenge, setting up a data lake is a solution suggested in both SLR and GLR. But SLR additionally gives an idea of how an Adapter pattern can be used to overcome this problem. The most discussed challenge in both SLR and GLR are *model monitoring* followed by *experiment tracking* and *Data access and management*.

The practical guidelines proposed based on the challenges and solutions are not tool-specific. Instead, they are general practices or ideas that can be used by the practitioners. However, the practitioner should take into consideration the type of incoming data, the complexity of the challenge(for example, a feature store may not be required in case of small ML projects), the tools that are feasible and compatible with the downstream application, and the costs before actually applying these guidelines.

Limitations

The blog postings, scientific papers, and tutorials are just a few examples of the systematic and grey literature materials we used in our research. The number of search terms and the chosen search engines may be insufficient for both SLR and GLR. Additional search terms and search engines might produce different or even better results. There are certain challenges like *Explainability*, *Coupling between system and model requirements*, *component entanglement*, *multitude of tools* for which solutions does not exist in our research. Finally, not all challenges have tool-specific solutions which led us to propose only a few guidelines for the challenges mentioned. Three guidelines were demonstrated out of seven and this is also not an evaluation of their effectiveness. The reason behind the demonstration of a few guidelines is, even though we use industry-relevant ML components for our demonstration, we do not have a real-world setting to get the real data (batch or streaming data) with data problems like inaccuracy, inconsistency, and ambiguity. This limitation will not provide us with sufficient conditions to demonstrate guidelines related to setting up a data lake, data labeling, and cleaning of training data.

8 Conclusion

This thesis highlights challenges in building the MLOps pipeline. Firstly, a systematic and grey literature review was performed to identify the challenges in different components of the MLOps pipeline, namely the data manipulation pipeline, model creation pipeline, and deployment pipeline. The challenges were classified into four categories as described above along with cross-cutting concerns. Based on this, seven practical and relevant guidelines were proposed and enriched with tool support. Three guidelines were demonstrated qualitatively by using them in the context of an example case with industry-relevant components.

The future work in this direction would be to find out more specific challenges with regards to different kinds of data(for example, batch data, streaming data) in the data manipulation pipeline, model creation pipeline, and deployment pipeline. Data is the most important entity in building ML applications. Different kinds of data pose different challenges. The next idea for future work would be to find ways to solve challenges for which solutions do not exist currently by expanding our study. The guidelines are validated in the local environment in our study, and validating these guidelines in realistic industry settings in the form of a case study provides a good scope for the future work.

9 Bibliography

[1] Alex Serban, Koen van der Blom, Holger Hoos, Joost Visser. *Adoption and E ects of Software Engineering Best Practices in Machine Learning*. URL: https://doi.org/10.1145/3382494.3410681.

[2] S. Alla and S. K. Adari, eds. *What Is MLOps?," in Beginning MLOps with MLFlow*. 2020th ed. Oreilly, Dec. 2020.

[3] Available on. URL: https://ml-ops.org/content/mlops-principles.

[4] Bin Qian, Zhenyu Wen, Devki Nanda Jha. *Orchestrating Development Lifecycle of Machine Learning Based IoT Applications:A Survey*. Nov. 2019. URL: https://arxiv.org/abs/1910.05433/.

[5] Ameet Talwalkar Book by Afshin Rostamizadeh and Mehryar Mohri, eds. *Foundations of Machine Learning*. 2nd ed. MIT Press, 2018.

[6] Cedric Renggli, Luka Rimanic, Nezihe Merve Gürel, Bojan Karlaš, Wentao Wu, Ce Zhang. *A Data Quality-Driven View of MLOps*. Feb. 2021. URL: https://doi.org/10.48550/arXiv.2102.07750.

[7] Patricia Cronin and Frances Ryan. *Undertaking a literature review: A step-by-step approach*. en. 10.12968/bjon.2008.17.1.28059. PubMed, Jan. 2008.

[8] D. Sato. *Thoughtworksinc/cd4ml-workshop: Repository with sample code and instructions for "continuous intelligence" and "continuous delivery for machine learning: Cd4ml" workshops*. URL: https://github.com/ThoughtWorksInc/cd4ml-workshop.

[9] *DVC*. URL: https://dvc.org/.

[10] *DVC*. URL: https://dvc.org/doc/user-guide/pipelines.

[11] *Evidently AI*. URL: https://docs.evidentlyai.com/.

[12] Floris Erich, Floris Erich, M Daneva. *A Qualitative Study of DevOps Usage in Practice*. June 2011. URL: https://10.1002/smr.1885.

[13] *Gaussian noise*. URL: https://www.sfu.ca/sonic-studio-webdav/handbook/Gaussian_Noise.html.

[14] Georgios Symeonidis, Evangelos Nerantzis, Apostolos Kazakis, George A. Papakostas. *MLOps – Definitions, Tools and Challenges*. Jan. 2022. URL: https://arxiv.org/abs/2201.00162.

[15] Geraldo Torres G. Neto, Wylliams B. Santos, Patricia Takako Endo, Roberta A. A. Fagundes. *Multivocal literature reviews in software engineering: Preliminary findings from a tertiary study*. en. 10.1109/ESEM.2019.8870142. IEE, Sept. 2019.

[16] *Git*. URL: https://www.simplilearn.com/tutorials/git-tutorial/what-is-git.

[17] *Heroku*. URL: https://www.heroku.com/about.

[18] *Introduction to KNN*. URL: https://www.ibm.com/topics/knn.

[19] Janis Klaise, Arnaud Van Looveren, Clive Cox, Giovanni Vacanti, Alexandru Coca. *Monitoring and explainability of models in production*. July 2020. URL: https://doi.org/10.48550/arXiv.2007.06299.

[20] *Kaggle Dataset*. URL: https://www.kaggle.com/datasets/mathchi/diabetes-data-set.

[21] Romesh Ranawana; Asoka S. Karunananda. *An Agile Software Development Life Cycle Model for Machine Learning Application Development*. en. 10.1109/SLAAI-ICAI54477.2021.9664736. 2021 5th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI), Dec. 2021.

[22] I. Weber L. Bass and L. Zhu, eds. *DevOps: A Software Architect's Perspective*. 2015th ed. Addison-Wesley Professional, 2015.

[23] Lucy Ellen Lwakatare, Ivica Crnkovic, Jan Bosch. *DevOps for AI – Challenges in Development of AI-enabled Applications*. en. https://ieeexplore.ieee.org/document/9238323. 2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Sept. 2020.

[24] M. M. John, H. H. Olsson, and J. Bosch. *Towards mlops: A framework and maturity model*. en. https://www.mdpi.com/2076-3417/11/19/8861/. Applied Sciences 2021, Sept. 2021.

[25] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, Martin Zinkevich. *Data Management Challenges in Production Machine Learning*. May 2017. URL: https://doi.org/10.1145/3035918.3054782.

[26] P. Ruf, M. Madan, C. Reich, and D. Ould-Abdeslam. *Demystifying mlops and presenting a recipe for the selection of open-source tools*. en. https://ieeexplore.ieee.org/document/9582569/. 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Sept. 2021.

[27] Joanna Olszewska Rex Black James Davenport, ed. *AI and Software Testing Foundation*. 1st ed. 2022.

[28] Rob Ashmore, Radu Calinescu, Colin Paterson. *Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges*. Oct. 2019. URL: https://doi.org/10.48550/arXiv.1905.04223.

[29] Ambler S, ed. *Disciplined agile delivery and collaborative DevOps*. 2011th ed. Cutter IT Journal, 2011.

[30] T. Granlund, A. Kopponen, V. Stirbu, L. Myllyaho, and T. Mikkonen. *Mlops challenges in multi-organization setup: Experiences from two real-world cases*. URL: https://oraviz.io/.

[31] M. Treveil and D. Team, eds. *Introducing mlops how to scale machine learning in the enterprise*. 2022nd ed. O'Reilly Media, Dec. 2022.

[32] V. Garousi, M. Felderer, and M. V. Mäntylä. *Guidelines for including grey literature and conducting multivocal literature reviews in software engineering*. en. 10.1016/j.infsof.2018.09.006. Information and Software Technology, vol. 106, Feb. 2019.

[33] *Visual Studio Code*. URL: https://code.visualstudio.com/docs/supporting/FAQ.

[34] C. Wohlin. *Guidelines for snowballing in systematic literature studies and a replication in software engineering*. en. http://dl.acm.org/citation.cfm?doid=2601248.2601268. Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14. the 18th International Conference. London, England, United Kingdom: ACM Press, Dec. 2014.

[35] Yue Zhou, Yue Yu, Bo Ding. *Towards MLOps: A Case Study of ML Pipeline Platform*. en. https://ieeexplore.ieee.org/document/9361315. International Conference on Artificial Intelligence and Computer Engineering, Oct. 2020.

A Appendix

Table A.1: MLOps Challenges and Solutions from Literature Review in Detail

MLOps Challenge	**Description**	**MLOps solution**	**Description**	**Tools**
Data access and management	It is a big challenge to locate data sources and comprehend their structure. The same data pipelines must be repeatedly created in the absence of a data catalog and dictionary. How can model runs be scaled, balanced, and parallelized for handling massive data streams of events with acceptable latency?	• Datalake • Adapter pattern		• Apache Flink • ONTAP • Apache Kafka
Shortage of diverse data samples	Class imbalance exist when models are not fed with all possible instances and types of data. They will start to malfunction when inputting unforeseen data in production and handling dynamic and shifting data is also necessary.	• Resampling • Synthetic data generation • Class weighting with data augmentation	• Up-sampling and down-sampling • The dataset includes artificial objects that are similar to the minority class. • The original data size is increased by applying label preserving changes.	

Continued on next page

Table A.1: MLOps Challenges and Solutions from Literature Review in Detail (Continued)

MLOps Challenge	**Description**	**MLOps solution**	**Description**	**Tools**
Data cleaning and validation	The data can be inaccurate, full of anomalies, and without proper data schema	• Data scrubbing • Verify that the data schema during training match with inference.	Data scrubbing is the process of altering or removing redundant, erroneous, incorrectly structured, or incomplete data from a dataset in order to increase consistency, correctness, and reliability.	• Alibi Detect • CPClean • ease.ml • Anodot
Data labeling	The labels are needed before the model can be retrained in supervised learning. Even tiny samples of data might be difficult to label since it takes time and requires subject-matter expertise.		The first option would be to label the new data using the trained model, or to utilize unsupervised learning rather than supervised learning, however this also depends on the nature of the issue and task's objectives.	• Labelbox • iMerit
Feature selection	It takes a lot of time to develop features from raw data. Every time a dataset is changed, the overhead increases even more. Costs for extracting the right features rise as the volume of data grows.	• Feature store • Enrichment	• ML functions are linked to a shared online and offline data repository and wrapped with metadata management. • It refers to the addition of new features to the training and serving data.	• Feast • Tecton
Calculation of performance metrics	Labels are required by metrics. They are always stateful by nature. Selecting the proper time frames for metric calculation is difficult.	• Minimum Functionality Tests • Invariance Tests • Directional Expectation Tests	It is customary to utilize label-independent or domain specific metrics as a proxy for model performance in the more prevalent circumstance of few labels.	• River • VMware vSAN Performance

Continued on next page

Table A.1: MLOps Challenges and Solutions from Literature Review in Detail (Continued)

MLOps Challenge	**Description**	**MLOps solution**	**Description**	**Tools**
Experiment tracking	Dealing with versioned data, parameters, hyper-parameters, ML models, and not just versioned code		Experiment tracking tools record all the essential information about the various experiments because ML projects need executing several experiments with various models, parameters, and training data.	• DVC • MLflow • Neptune.ai • CML
Algorithm and hyper-parameter selection	One of the most time-consuming and resource-intensive activities in an ML workflow is tuning the parameters, hyper-parameters, and choosing the best algorithm.	Automl	AutoML workflow includes data preparation, model development, hyper parameter adjustment, assessment, and validation. A number of models are trained on the same data set with automl and the best model is exported.	Optuna
Model evaluation	Overfitting results from continuous testing. Compared to other software systems, testing an ML system also presents more operationally challenging regions because it incorporates data, models, and code.	• Regularization • Cross validation • Dropout	Deepchecks is open source package that includes comprehensive test suites that are adaptable, expandable, and editable for machine learning models and data.	ease.ml
Model monitoring	Monitoring should capture data drift to avoid training-serving skew. Identifying the feature that causes the drift is crucial to the performance of the model.	• Model picker	Model Picker is an online model selection approach in which poorly performing models are retired while new or standby models are brought in.	• Amazon SageMaker Model Monitor • Evidently AI • MLWatcher

Continued on next page

Table A.1: MLOps Challenges and Solutions from Literature Review in Detail (Continued)

MLOps Challenge	**Description**	**MLOps solution**	**Description**	**Tools**
Managing deployment pipelines	Due to the iterative nature of machine learning, models usually need to be retrained and validated before being used in production. All of these deployment processes if done manually, takes a lot of time and effort.	Efficient automated pipelines right from the development	Deploy quickly through the use of automated CI/CD workflows. All three pipelines—the data manipulation, model building, and deployment pipelines should be compatible with one another.	• Kubeflow • TensorFlow Extended
Operations and feedback loops	During model degradation its difficult to decide the threshold for triggering model retraining, replacement of model without stopping the production, and deciding on what information needs to be sent from deployment pipeline.			
Incompatibilities between development and production	Incompatibilities in code change, dependencies,build script, and relying on notebooks in development environment for building models.	• Staging integration test • Canary integration test • Production rollout	• To verify basic functionalities • To ensure serving performance across all production models • Modular code easily testable and movable to production	
Vendor lockin with commercial MLOps platform	Due to complexities in computing resources and software frameworks, ML projects are bundled with commercial cloud platforms referred as vendor lockin.	Devise a MLOps platform with open source frameworks		Kubeflow

Continued on next page

Table A.1: MLOps Challenges and Solutions from Literature Review in Detail (Continued)

MLOps Challenge	**Description**	**MLOps solution**	**Description**	**Tools**
Difficulty to estimate infrastructure resources	ML models require high-performing compute resources like GPUs which is difficult to estimate due to potentially voluminous and varying data during training and serving.	Automatic scaling	Training and serving pipeline should be capable of automatically scaling resources. Reuse of code and model reduces usage of resources.	• NVIDIA DGX • MLib
Explainability	The MLOps system should have the power to observe and act on insights with self-explanatory capabilities, justifying why a model behaved in a particular way.			Seldon
Security and privacy	ML models are frequently a part of larger systems whose output is used by unidentified applications raises a number of security concerns. Navigating the trade-off between privacy and utility is seen as an open challenge.	Security and access control	To guarantee that only authorized users can use the outputs of ML models by adding automated security and compliance testing to data pipelines to make sure the data is safe and compliant.	
Forgetting the downstream application of a new model	There is no assurance that the data output from model will be in a format that the business API can utilize if this downstream application is not taken into account when the model is constructed.	A great way to avoid this is by creating a walking skeleton or steel thread		

Continued on next page

Table A.1: MLOps Challenges and Solutions from Literature Review in Detail (Continued)

MLOps Challenge	**Description**	**MLOps solution**	**Description**	**Tools**
Coupling between system and model requirements	Dataset features and system features may be connected. What characteristics of ML models should be mentioned when a feature is specified as a system requirement and what are the performance effects on each other?			
Component entanglement	The models can interact in unanticipated ways during development, making it challenging to maintain clear boundaries between them.It may result in non-monotonic error propagation. It is challenging to identify and troubleshoot hidden feedback loops caused by dependencies between data or between models.			

Continued on next page

Table A.1: MLOps Challenges and Solutions from Literature Review in Detail (Continued)

MLOps Challenge	**Description**	**MLOps solution**	**Description**	**Tools**
Multitude of tools	ML developers frequently want to test every tool (e.g.,algorithm), experiment with various data preparation libraries (e.g.,Pandas and Apache Spark), various model types (e.g.,trees and deep learning), to determine if it enhances performance. Many tools are bulky and labor-intensive to deploy and use. For maximum effectiveness, there isn't a single set of monitoring tools or pipelines that "follows around" each data engineer.			

Table A.2: References of SLR and GLR

Source	**Link**
Reference of SLR	https://zenodo.org/deposit/7103232
Reference of GLR	https://zenodo.org/deposit/7103232

DECLARATION

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature