CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Applying MLOps to a Data Visualization System: A Case Study

Master's thesis in Computer science and engineering

Wei Guo
Wenjie Jiang

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Applying MLOps to a Data Visualization System: A Case Study

Wei Guo
Wenjie Jiang

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Applying MLOps to a Data Visualization System: A Case Study
Wei Guo, Wenjie Jiang

Supervisor: Daniel Strüber, Department of Computer Science and Engineering
Advisor: Gordon Sun, Software by Quokka AB
Examiner: Gregory Gay, Department of Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Applying MLOps to a Data Visualization System: A Case Study
Wei Guo, Wenjie Jiang
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis investigates the integration of Machine Learning Operations (MLOps) in the development of a data visualization system, aiming to assess its impact on data quality, code quality, and model quality. The study addressed two primary research questions: (1) To what extent can MLOps be helpful in a data visualization system? and (2) Which best practices can be derived from applying MLOps to such a visualization system? To address these questions, a case study was conducted to investigate if there was an improvement when introducing MLOps to the system. Through a combination of experimental and observational work, the research analysed MLOps improvement from both quantitative and qualitative aspects, and identified best practices from the development of the case study. These findings contributed to bridging the gap between different MLOps applications in data visualization field, and served as a successful example for practitioners, developers, and researchers in the intersection of MLOps and data visualization.

Keywords: Machine learning Ops; Machine learning; Data visualization; DevOps; Case study

Acknowledgements

This thesis work was done in collaboration with Software by Quokka, Chalmers University, and Gothenburg University. The authors would like to thank the supervisor, Dr.Daniel Strüber, for his invaluable technical support and guidance, mentor Gordon Sun, for continuously raising the feasible requirements in each phase, and the examiner, Dr.Gregory Gay, for his comments.
The authors would like to thank the technical guiders: Mentor Simon Cao for AWS environment support and Rex Ruan for his development and environment configuration support. Special thanks to Marina Duan for proofreading. Thanks to SiSi Lai and Chewchew for the emotional support. The authors would also like to thank all the testing participants involved in the maintenance and feature experiments.

Wei Guo & Wenjie Jiang, Gothenburg, March, 2024

List of Acronyms

Amazon EC2 Amazon Elastic Compute Cloud. 30, 41
Amazon RDS Amazon Relational Database Service. 30, 41
Amazon S3 Amazon Simple Storage Service. 30, 32, 41
API Application Programming Interface. 32
AWS Amazon Web Services. 28, 30, 33, 40, 41

CD Continuous Delivery. 6, 13, 18, 22, 34, 38, 48
CI Continuous Integration. 6, 13, 18, 22, 34, 38, 48
CT Continuous Training. 20, 41

ETL Extract, Transform, Load. 20

IT Information Technology. 5

ML Machine Learning. 1, 2, 5, 6, 25, 27, 28, 34
MLOps Machine Learning Operations. v, xi, xiii, 1–6, 8–11, 18, 25, 27–31, 33–35, 37–49, 51, 53

NER Named-entity Recognition. 16

Contents

List of Figures

List of Tables

1

Introduction

The most recent trend in the software industry centres on ingratiating software with Machine Learning (ML). Some popular applications, such as Bing Search [1] and Siri [2] voice assistant, have primarily benefited from the breakthroughs in ML. The transition from traditional software to ML-based software spurs software organizations to evolve new development practices which can adapt ML smoothly [3]. One of the development practices is Machine Learning Operations (MLOps). This thesis analyzed whether MLOps is helpful to the data visualization system from multiple perspectives, and how much it helps.

1.1 Problem Description

Machine Learning Operations (MLOps) is an emerging paradigm which combines DevOps with the unique needs of machine learning. DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) aimed at shortening the development life cycle and providing continuous delivery with high software quality [4]. Similarly, MLOps aims to deploy and maintain machine learning models in production reliably and efficiently. This promising approach can improve model performance, respond to growing better datasets, and ease maintenance workload [5]. As an illustration, Google's autocorrect continuously improves because MLOps pipelines repeat all ML operations on fresh data to automate and refine the model [6].

With the rise of ML technology in the industry, data has become vital in the different applications and has received increasing attention. Consequently, there is a rising trend in effectively processing, visualizing, and analyzing data in industry and academia. Data visualization, as the graphical representation of data, has come to the public's attention and gained a lot of influence. Because it provides users with a more intuitive way to demonstrate the results of data analysis. The rapid prosperity of ML has also upsurged in data visualization systems to mine more valuable information from data. However, involving ML in data visualization systems can fall into the category of dealing with constant changes arising from continuous supply of new data. Especially, in the data visualization field, the dataset is quite big. If the machine learning model can not be updated continuously, the change in the dataset will finally lead to outdated machine learning models and data analysis results.

As a comparison, MLOps automate, develop, and deploy ML models across the whole journey of software applications. MLOps pipelines can help with data visualization systems to adapt to new data continuously without much manual intervention. In addition, in the highly competitive market, data visualization systems have high demands for data timeliness. When new data becomes available, the expeditiousness of visualizing analyzed results is crucial for gaining a competitive position in the market. The celerity of MLOps is extremely impressive in automating the entire development pipeline, guaranteeing delivery on time, and reducing engineers' workload.

Unfortunately, many data visualization systems neglect the incorporation of MLOps, resulting in lower model accuracy on a large scale [7]. Also, a lot of time and money is spent to regularly retrain the ML model. Consequently, there is a question regarding MLOps efficiency and accountable benefits. The uncertainty led to MLOps limited adoption. This thesis examined to what extent MLOps could help a data visualization system and explored the potential of MLOps to provide best practices for a visualization system.

1.2 Purpose of study

This thesis presents a case study of applying MLOps within a data visualization system in collaboration with Software by Quokka and aims to assess the impact of MLOps in the data visualization field. Overall, this thesis research addressed two main research questions:

- **RQ1**: To what extent can MLOps be helpful during a data visualization system development?
  - RQ1.1: To what extent do MLOps help improve data quality?
  - RQ1.2: To what extent do MLOps help improve code quality?
  - RQ1.3: To what extent do MLOps help improve model quality?

Due to the lack of existing MLOps applications in data visualization field, the research also aims to provide a plausible successful example for practitioners, developers, and researchers in the future.

- **RQ 2**: Which best practices can be derived from applying MLOps to such a visualization system?

Furthermore, to assess the actual impact of MLOps, a ML based visualization system was built. In this thesis, the company sponsored to develop the system. According to the company's requirements, this thesis addressed the following machine learning problems in this specifically developed system.

- Location parsing: analyze the geographical origins of institutions in the bibli-

ography information of research papers.

- Author disambiguation: resolve the issue of identifying authors with similar names when multiple papers are involved.

These two machine learning problems would be comprehensively explained in chapter 2. Moreover, these two machine learning problems would be involved in the case study for MLOps practices in data visualization systems

1.3 Significance of the Study

The significance of this research is that it contributes to understanding the role of MLOps in data visualization systems, which is an area worth exploring. Data visualization systems, which often deal with changing data, tend to face the problem of machine learning models becoming outdated more than other areas. These systems also require automated training, integration, and delivery more than other fields because the timeliness of data is critical to visualization systems.

In addition, this study also evaluates MLOps effectiveness from a qualitative and quantitative perspective and combines academic and commercial perspectives to make evaluation metrics, which can be used as an evaluation metrics reference for future research on MLOps helpfulness.

By highlighting the challenges and opportunities associated with the integration of MLOps, this article aims to provide valuable guidance to practitioners and researchers in the field and define core best practices for future attempts in similar areas. The expected results of the study will highlight the importance of MLOps in keeping data visualization systems relevant and effective, which helps in making better decisions and encourages ongoing improvement.

1.4 Thesis outline

The entire thesis is divided into six chapters.

- Chapter 1 introduces the project background and discusses the scope of the thesis.

- Chapter 2 highlights the core theory of MLOps, states the current state-of-the-art, and elaborates two underlying machine learning focuses.

- Chapter 3 illustrates the system which was developed without MLOps and comprises the integration of MLOps with the data visualization system.

- Chapter 4 explores the research questions and corresponding methodology, and the criteria used to measure the benefits of applying MLOps to the system.

- Chapter 5 reviews the development process and generalizes the valuable con-

tributions of integrating MLOps to data visualization systems from multiple perspectives. This chapter also covers the validity of threats.

- Chapter 6 concludes the comprehensive discussion about the findings, summarizes the entire thesis, and provides guidance into potential future studies.

2

Background

The thesis dives into how MLOps can help build a data visualization system, seek a solution through combining MLOps with the system, and collect practices by applying MLOps during development. It also highlights the critical role of MLOps in the lifecycle of ML models and the benefits which MLOps offers in automating and managing these systems. This chapter provides an in-depth knowledge of MLOps, compares various MLOps tools, and discusses their applications in data visualization.

2.1 MLOps fundamentals

2.1.1 MLOps basic theory and What MLOps include

MLOps is an emerging discipline at the intersection of ML, DevOps, and data engineering. It fosters collaboration and communication between data scientists and Information Technology (IT) professionals while automating ML algorithm deployment and prediction processes [8].

DevOps is a "set of practices and tools focused on software and systems engineering". It integrates and automates software development work that represents "Dev" in DevOps, and IT operations work, encompassing "Ops" in DevOps to enhance and shorten the system development cycle [4]. For example, with DevOps, development teams build software by incorporating testing early in the development process. Software integration tests can be automatically executed after each commit by using Jenkins, a widely used tool for continuous integration.

Like DevOps, MLOps aims to establish a culture and environment where ML technologies can generate business benefits rapidly, frequently, and reliably build, test, and release into production via practices and tools. The complete MLOps process includes three broad phases: **"Designing the ML-powered application," "ML Experimentation and Development," and "ML Operations"** [9]. The initial stages, namely "Designing the ML-powered application" and "ML Experimentation and Development," centre around designing and developing machine learning models which are tailored to meet the requirements of the products and customers. The third phase, known as "ML Operations," serves as the core step within the MLOps framework and encompasses the following substeps as outlined in the literature [9]:

- Continuous Integration: This process extends the testing and validation of code and components to include the testing and validation of data and models.
- Continuous Delivery: This involves the creation and maintenance of an ML training pipeline which automatically deploys a subsequent ML model prediction service.
- Continuous Training: This is a property unique to ML systems which automatically re-trains ML models for re-deployment.
- Continuous Monitoring: This refers to the ongoing monitoring of production data and model performance metrics, which are linked to business metrics.

Delivering an ML model to production involves several steps: **data extraction, data analysis, data preparation, model training, model evaluation, model serving, and model monitoring**. These steps could be completed manually or automatically. Based on the number of automation steps, the MLOps can be divided into three levels based on the scientist's definition. In the first level, all of these processes are performed manually. The machine learning pipeline is automated in the second level, but the connection with the software is made by engineers. In the third level, the machine learning part and the whole system would benefit from the Continuous Integration (CI)/Continuous Delivery (CD) and automated machine learning pipelines.

Building the third-level MLOps process is the main focus of this thesis. **Figure 2.1** gives an overview of the data process for the whole MLOps pipeline, and combines the MLOps steps with the machine learning system development steps. At the top of this figure, the three grey blocks define the development steps of an ML-based system, from simple codes to real applications. Between them, the blue arrows signify the different stages of machine learning operations. Through continuous integration, simple codes become packages or artifacts. Then, a continuous deployment pipeline automatically deploys the packages to the required environment and runs tests there. The packages are automatically deployed to the production environment if everything works fine.

In particular, the development stage includes regular front-end coding, back-end coding, and the development of machine learning models. Furthermore, when new data comes in, the MLOps pipeline automatically triggers the data processing and then trains the machine learning model based on the latest data. At the same time, MLOps supports monitor features. The monitoring tool can continuously monitor the performance of a machine-learning model. If the performance is lower than the baseline, it can automatically revert to the previous working machine learning model or alert data scientists. Moreover, the new model goes through the CI/CD pipeline to the production environment with new data.

Figure 2.1: The Third-level MLOps Pipeline

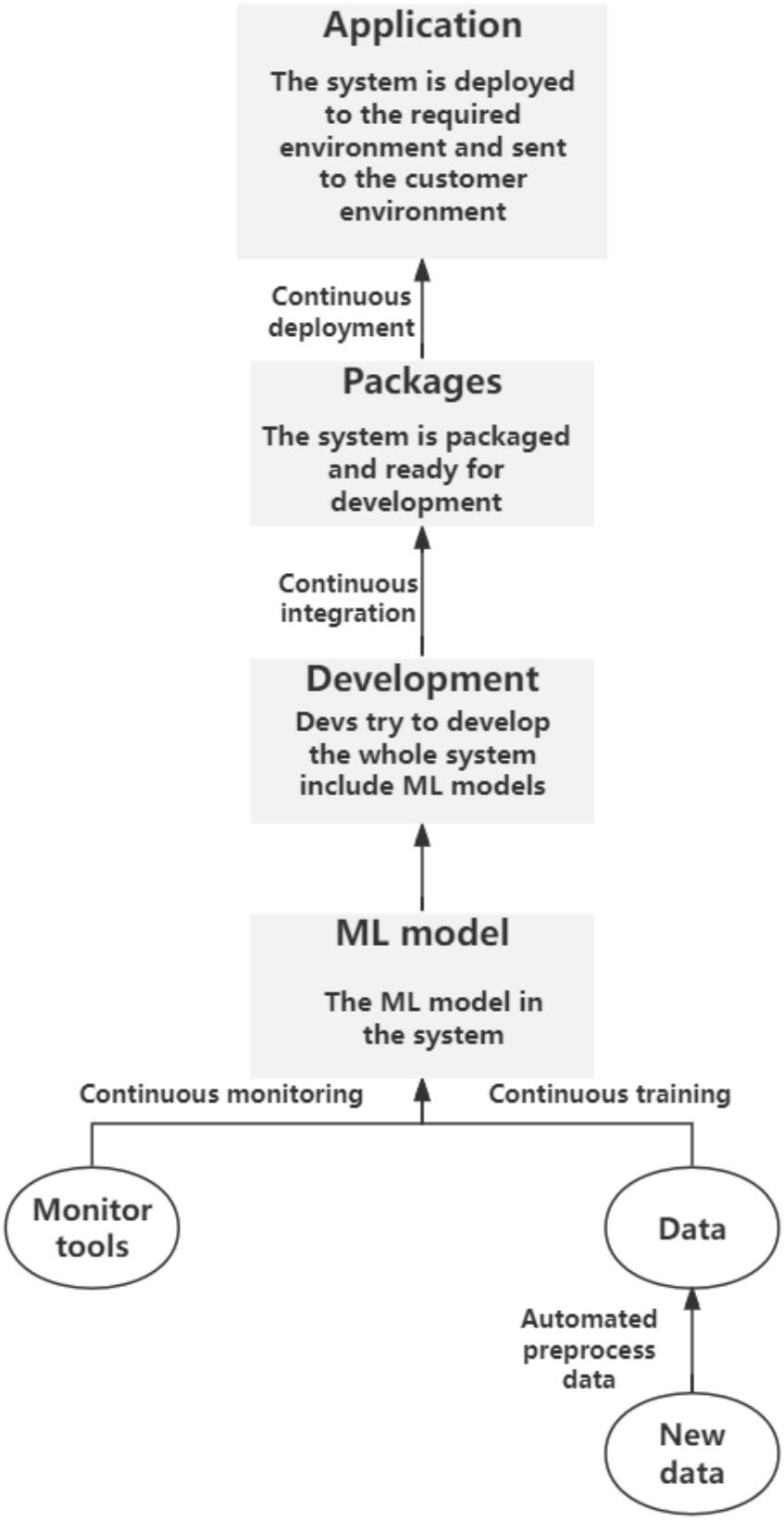


Figure 2.2: ML flow Pipeline

2.2 MLOps tools and comparison

Although MLOps is a new field, there are still some relatively mature tools available. These MLOps tools enable easy, reliable and fast management of the ML lifecycle [10]. For example, ML flow is a classic open-sourced MLOps tool.

This section introduces the fundamentals of different MLOps tools and their differences. Moreover, the chosen tool for this data visualization system is introduced, and the reason behind will be discussed.

2.2.1 ML flow

ML flow is an open-source platform for the machine learning lifecycle. It can manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry [11]. The ML flow pipeline is presented in Figure 2.2.

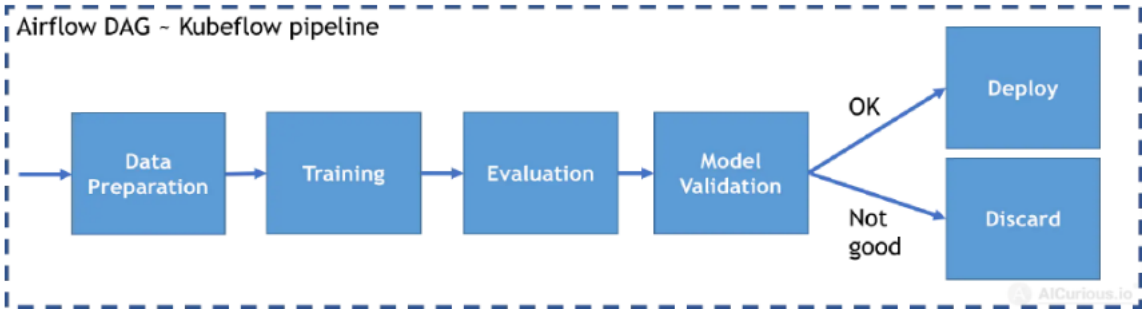
In the first part, experiment tracking, ML flow records and queries experiments: code, data, config, and results based on the developers' requirements. Furthermore, ML flow would register the model after the experiment and store, annotate, discover, and manage models. Moreover, ML flow also has the feature of deploying machine learning models in diverse serving environments [11].

2.2.2 Airflow

Airflow is also a popular open-source tool in the MLOps fields. It is a community-created platform for programmatically authoring, scheduling, and monitoring workflows [12].

The overall pipeline is shown in Figure 2.3 [13]. The figure shows that the main difference between ML flow and Airflow is that Airflow includes more steps, while ML flow focuses more on machine learning models. Airflow has good support for data preprocessing. It has the features of data version storage and rollback. Moreover, as a workflow orchestration framework, it supports the machine learning lifecycle and the entire project life cycle. Nevertheless, Airflow cannot provide registration and multiangle monitoring functions for machine learning models. It means that the project cycle covered by Airflow is longer than ML flow, however, for the machine learning models lifecycle, the support is not as good as that of ML flow.

Figure 2.3: Airflow Pipeline



2.2.3 Other platforms

In addition to the two classic tools mentioned above, now, with the gradual application of MLOps in the industry, more tools are emerging in the market, such as the Kubeflow project launched by Google dedicated to deploying ML models, which allows Kubernetes to manage simple, portable, and scalable deployments on demand [14].

Based on Ml flow, Kubeflow uses Kubernetes as the underlying tool to carry the entire pipeline and has made significant progress in stability and scalability. However, it comes with the rise of learning difficulty and the disadvantage of complex construction. For small projects, it is not suitable.

In addition, considering that data scientists may not have a solid computer science background, some MLOps tools on the market simplify code flow, such as Neptune.ai [15], DataRobot [**16**]. Using these tools to build an end-to-end tool is more efficient than using ML flow and Airflow since it saves a lot of shell scripts and DevOps code. However, because these tools simplify many processes, the degree of freedom of the entire project process is low. It must be carried out according to the officially recommended process. Otherwise, it is challenging to build the entire pipeline. The lack of freedom is also one of the drawbacks of these tools.

2.2.4 Summary

Considering the development cost of the system, the degree of freedom, and the degree of support for the company's platform, the ML flow was ultimately chosen. Firstly, as a relatively classic tool in MLOps, the ML flow supports various languages and plugins very well. Secondly, the entire code is open source and developers can change and adjust it according to the project needs. Thirdly, compared to ML flow, Airflow supports more functions. The learning cost is relatively high and requires stricter project process requirements. Therefore, the ML flow was chosen to build the platform in our project. The content of specific projects is introduced in Chapter 3.

2.3 MLOps helpfulness

Creating machine learning solutions to various problems can be a challenging task including many stages, such as collecting and processing raw data, analyzing the data, constructing, training, and testing the model. Furthermore, in the case of advanced machine learning software that continuously receives new data, these steps need to be iterated multiple times. While the process becomes simpler during subsequent iterations as developers only need to refine the model based on new data patterns and trends, it still poses a challenge that may require hours of manual work that could be more effectively utilized elsewhere.

With the assistance of MLOps, developers can take full advantage of high-performance machine learning models without the hassle of applying some other pipelines. This is where MLOps comes in, which can be considered as the intersection between machine learning and DevOps practices [16].

Before the development of MLOps principles, the large quantity of resources needed to implement solutions, which made employing the most recent machine learning technologies a substantial issue for enterprises. However, with MLOps technology coming up, automating the most complicated components dramatically speeds up the development and maintenance processes, making the deployment and maintenance of machine learning systems considerably simpler. With a completely automated setup, developers can stay up-to-date with the latest machine learning technologies and swiftly deploy new models. As companies install more unique and more promising model architectures, services may retain their high level of performance and possibly enhance them [16].

ParallelM, a company in the machine learning field, believes that this discipline, MLOps, which draws on previous models like DevOps but expands to address unique machine learning needs, is critical for businesses to close the loop and generate highly adaptive and competitive ML business applications [17]. Overall, it promotes automation, versioning, explainability, and traceability.

2.4 The existing ML-based data visualization systems

This thesis examines the use of MLOps in a data visualization system based on machine learning. As a result, some commonly used machine learning based data visualization solutions are investigated, such as Deepeye, DeepDrawing, GenerativeMap, HDBSCAN, and BIBLIOMETRIX.

- Deepeye: A novel system for automatic data visualization. It applies machine learning technology to address how data visualization can be improved [18].
- DeepDrawing: Similar to Deepeye, it also uses machine learning to help users have better data visualization results. However, the difference is that Deep

Drawing focuses on visualization improvement, which means the user already has the basic figure. DeepDrawing focuses more on studying the style of previous visualizations and directly mocks new visualization figures for users [19].

- GenerativeMap: This system uses machine learning to read the density map (a density map widely used for data sampling, time-varying detection, ensemble representation, etc.) and generate a better one that can give more intuitive results to the authors [20].

Machine learning is also applied in the bibliographic information systems field. For example, in a system, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) was applied to the clustering prediction of the author's dominance ranking [21]. Furthermore, BIBLIOMETRIX is a famous tool for quantitative research in scientometrics and bibliometrics that includes all the main bibliometric analysis methods. It also applies the machine learning technique to more comprehensively analyze bibliographic information [22].

In total, the investigation examined more than 30 machine learning-based data visualization systems. They are spread across fields, but none of them have MLOps applied. On the basis of the research, the authors believe that if these systems are designed to last a long time, they probably run into the issue of the model expiring as the data changes. These require to be manually adjusted frequently. Although, if the system is being built for research purposes, it would be logical to put it on hold.

2.5 Related work

In this research, the initial stages of MLOps practices are explored, highlighting that the field is relatively under-researched compared to DevOps. This is particularly true within the context of data visualization systems, where the dynamic nature of data and the need for timely model updates present unique challenges.

2.5.1 MLOps application in similar area

With the increase in machine learning projects, MLOps has also been applied widely in practical scenarios. According to Sasu Mäkinen's research, organizations across 63 countries use MLOps. In particular, their work focuses on how to use data better through MLOps and using MLOps to build the first batch of models and puts them into production [23].

The study draws upon existing applications of MLOps in similar systems, such as those used in the electricity market with similar large datasets and data visualization charts [24]. In the case study, MLOps pipelines have been employed to predict market prices and adjust national electricity demand forecasts. The study emphasizes the importance of market timings in model predictions and the use of MLflow for managing the ML application lifecycle and highlights how MLOps can reduce

manual retraining and decrease human intervention.

There are also MLOps applications in IoT systems. For example, one study collaborated with the Finnish Technical Research Center and is interested in forecasting air quality in real time. Similarly to the data visualization field, this kind of system has a high demand for data timeliness. The study aims to adapt Edge MLOps to a forecasting air quality system and investigate how much it can help with cost, energy and operational efficiency [25].

However, although these studies point out the benefits of MLOps in areas similar to data visualization, they are either very general qualitative analyses without sufficient experimental data support, or they focus more on edge MLOps vs cloud MLOps comparison but little research on the benefits of MLOps itself [25]. In contrast, this research is more focused on the benefits brought by MLOps itself and the benefit is divided into different parts for discussion, which is more detailed than the previous general research. Furthermore, the benefit is evaluated from both qualitative and quantitative perspectives and experiments will be conducted to make the conclusions more effective.

2.5.2 Improvement of data visualization systems

Information visualization technology aims to make graphical representations easier and faster for humans to understand, helping users absorb the inherent behaviour of data and identify relationships between data elements [26]. With the rise of machine learning technology, data visualization systems have also adopted machine learning technology as a key step in data analysis to show users more relationships between different data elements.

However, with the development of the Internet, the data that can be collected has increased exponentially, and the size of the data sets managed by information systems has become larger and larger and is usually multi-dimensional. This brings considerable challenges to the data visualization system in processing data. Some existing research mainly starts from the data visualization system itself, such as strengthening the management of the database and introducing more SQL statements to speed up data operations in the system. Or use approximate solutions and asymptotic solutions to deal with big data to provide real-time responses [27].

Nevertheless, many visualization systems still face other challenges after integrating many database algorithms. For example, a study on medical visualization systems pointed out that currently, medical knowledge is mainly integrated through manual compilation of guidelines into Clinical Decision Support (CDS). Previous work has shown that these guidelines are not always followed and may become outdated [28]. In addition, many visualization systems incorporate machine learning algorithms to find the most appropriate chart to display to users from a large number of chart forms. But visualization is essentially a human-centred field. Human involvement is critical to the success of visualization [29]. Existing machine learning visualization systems lack human feedback, leading to users' confusion.

MLOps has demonstrated the potential of automated machine learning workflows in other areas to simplify and optimize processes [30]. Furthermore, efficient database management techniques can be combined with MLOps practices such as CI/CD pipelines for machine learning models, which can also ease the process. In addition, incorporating feedback loops into automated workflows is also in line with the concept of MLOps and can ensure that the data visualization system remains intuitive, relevant and meets user needs. Moreover, because of its automation support, MLOps can automate model training and data analysis when new data arrives, which can satisfy the needs of data visualization systems. Key fields such as healthcare especially need high adaptability to real-time data analysis and data visualization needs.

Overall, based on the information above, we infer that combining advances in data visualization with MLOps practices provides a comprehensive approach to developing efficient data visualization systems. However, little research has investigated whether MLOps can help with the data visualization system. This research extends the MLOps findings by addressing the mystery of MLOps benefits in the data visualization field.

3

Context

This thesis was written in cooperation with a company, Software by Quokka, worked with a project about a machine learning based data visualization system and applied MLOps to it. Quokka provided different technical support, including assigning senior DevOps engineers and senior ML engineers as advisors. Additionally, the company offered specific financial support, including funds to rent servers and working laptops. The development carried out with AWS would guarantee the feasibility of the thesis research study to a certain degree.

This chapter describes how this ML-based data visualization system was developed and introduces how MLOps was applied. The development process was related to the methodology for solving the research questions.

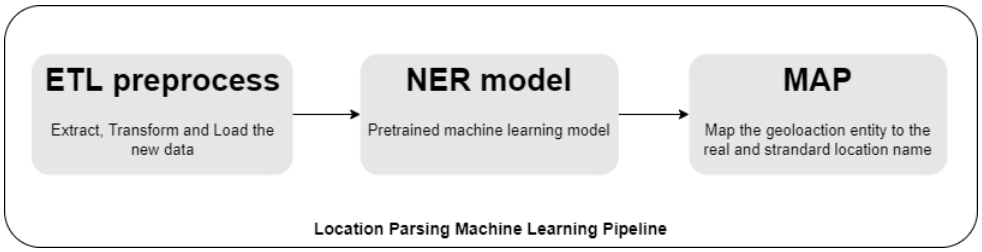
3.1 The data visualization system without MLOps

The idea of the machine learning based data visualization system which the company offered is to read bibliographic information to build one city and author ranking system. The system helps students, researchers, and practitioners get intuitive visualizations and up-to-date rankings results.

The system requires specific and accurate author institution addresses for the city ranking feature to get accurate results. The data source for this project comes from a website called PubMed that comprises more than 33 million citations for MEDLINE biomedical literature, life science journals, and online books [31]. The data has a long history, some are old and could be traced back more than 30 years. At that time, there was no standardized format to follow. The institutions' addresses of many papers are not standardized, and the accuracy of extracting with simple keywords is not high. Therefore, the data visulization system must apply natural language processing technology to parse the institution's address string to obtain a more accurate author address. As a result, location parsing becomes the first machine learning problem.

Furthermore, for the author ranking feature, it requires adequate author identification. However, because different authors may have the same name (homonyms), one author may publish under a different name (synonym) [32]. Thus, author dis-

Figure 3.1: Location Parsing Machine Learning Pipeline



ambiguation also becomes the second machine learning problem.

This chapter mainly introduces how to solve two machine learning problems in the system and rationally discusses this solution. After the development, the system became the basic case example for this thesis. In particular, because this paper focuses on applying MLOps when solving machine learning problems, the existing solutions were used as much as possible meanwhile focusing on applying MLOps.

3.1.1 Location parsing

Extracting correct geographic information from significant texts has always been an essential topic in natural language processing. There are applications in medical and health fields, such as Carmen (A Twitter Geolocation System with Applications to Public Health) [33]. Many people are studying this topic, so many open-source machine learning geographic location analysis tools are available on the Internet.

Through research including resources in Github and papers online, the general idea of these tools is to first extract all geographic location entities in the text through the entity extraction model. Then the local geographic information database or the online database is used to retrieve geographic location entities. Based on the returned results, it determines whether the entity is a definite geographic location or not. Alternatively, it determines where the entity is more likely to correspond to the actual geographic location in the case of ambiguity. Furthermore, the results are in a standard format for geolocation.

Due to the large amount of data provided by the company, the existing location parsing tools are not suitable, for example, mordecai [34] and GEOparse [35]. The processing time is often too long, so the authors created their own pipeline based on the above ideas. The pipeline is shown in Figure 3.1.

In the first part of the figure, a Python file was used to prepossess the data. In Named-entity Recognition model part, the entity extraction pipeline was used which is provided by spark NLP in the entity extraction part. Spark NLP was chosen [36] because it has better support for big data as a spark native machine learning library. In particular, the system, as a data visualization system, deals inevitably with millions of data. Spark NLP shows better performance than other entity extraction warehouses with this level of data [37].

In the data mapping part, Geonames, an open source local database [38], was used. In addition, elastic search was used to package it. To some extent, the elastic search ensures that in the case of large data throughput, the database can still maintain a relatively high-level search speed.

After implementation, the machine learning model was briefly evaluated. Considering that the system's input data was unlabelled data, some data was marked during the experiment to test the accuracy. However, the disadvantage of manual marking was that results could be more serendipitous duo to the small amount of data. Therefore, the open-sourced data was used in Kaggle [39] as a reference for the accuracy test.

3.1.2 Author disambiguation

Name ambiguity in the context of bibliographic citation records is a complex issue that affects the quality of services and content in digital libraries and similar systems. Handling name ambiguity was essential for accurate results in this bibliographic data visualization system.

According to the research, many mature methods to solve the problem, including supervised similarity learning through clustering, supervised similarity learning through classification, graph-based similarity generation through clustering, and heuristic-based approaches [40].

Considering the amount of data this system processes is significant, and name disambiguation is a general machine learning problem, many open-source data files on the Internet can be used for training and testing. It was finally decided that the supervised learning method would be used.

Among the supervised methods, the Random Forest classifier proposed by Kaushal Jhawar [41] has entered the field of vision with its straightforward construction and environmental requirements. In this solution, the raw data is first processed to calculate the similarity of each other's institutions, co-author names, and journals. Among authors with the same name, the similarity data is loaded into a random forest classifier to judge it. The general pipeline is shown in Figure 3.2.

Considering how well the model matches the system, it was reconstructed based on its model. Some unimportant features were removed, and some other features were optimized to calculate the similarity. The final model was the one used in this data visualization system.

3.1.3 Overall system design

Now there are two trained machine learning models regarding the problems to be solved, which would to be integrated to build an end-to-end system. The overall system design is shown in Figure 3.3. Some functionalities were created, including extracting bibliographic information from an open-sourced online paper website and

analyzing with the machine learning models. Then the result is pushed to a simple frontend to visualize the city and author rankings based on the number of published papers.

3.2 The data visualization system with MLOps

Since the system was ready, the next step became how to apply MLOps on it to get the result of how MLOps influenced the system performance, which was one of the research questions. Because the development process was related to the methodology of solving the research questions and had some effects on how to evaluate the results, the details of the example are introduced here.

As mentioned in Chapter 2, the overall MLOps process consists of three parts: "Designing ML-Driven Applications," "ML Experimentation and Development," and "ML Operations." The first two steps are completed. For the third step, it was broken down into machine learning pipeline automation and CI/CD pipeline for the entire system.

3.2.1 ML pipeline automation

Machine learning pipeline automation is one of the sub-steps of MLOps. It is designed to automatically train models through an automated pipeline for continuous delivery of machine learning. According to Figure 3.4 in Chapter 2, its flow is as follows. The new data comes in automatically when the source website updates, and it will be dealt with by the data pipeline. Then, it triggers the continuous training pipeline and continuous monitoring pipeline working at the same time, the ML model would be updated if the newly trained model's overall performance is better than the current model.

Figure 3.2: Author disambiguation machine learning model

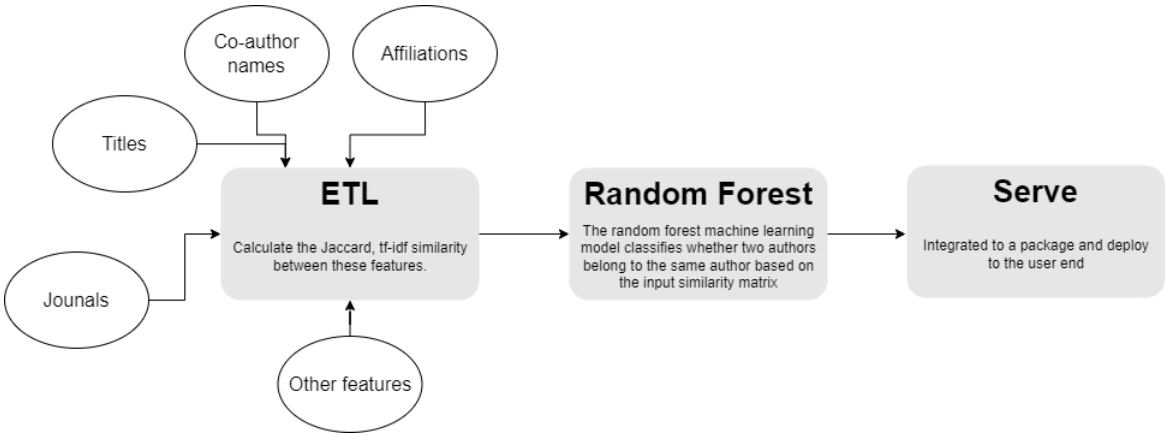


Figure 3.3: Visualization system without MLOps

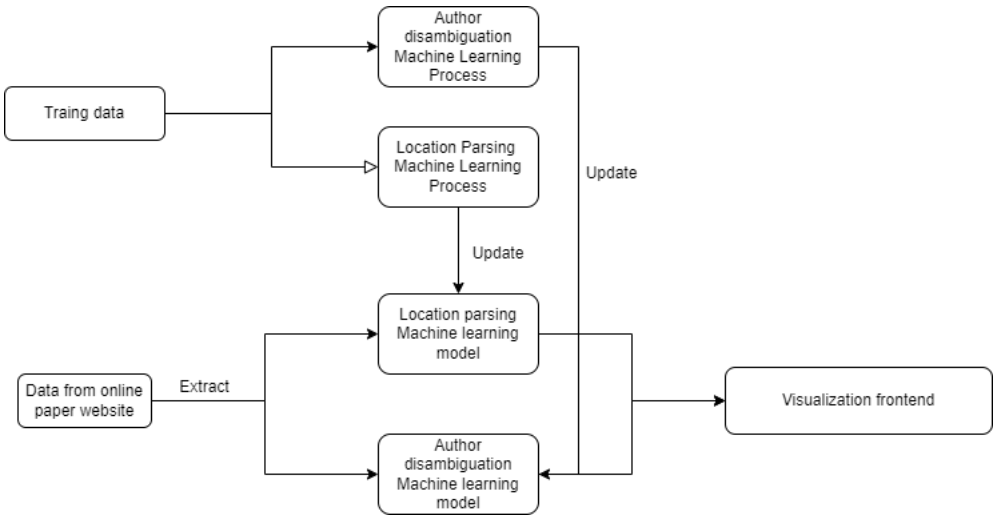
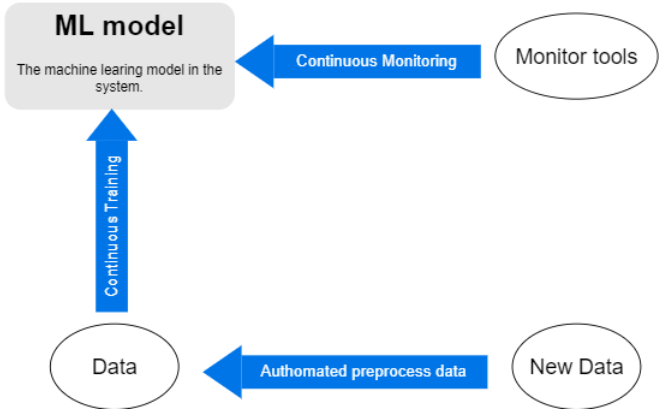


Figure 3.4: ML pipeline automation



3.2.1.1 Location Parsing

According to the design of the location parsing machine learning model, there is a location parsing machine learning automation pipeline as shown in the Figure 3.5. Since the system has data updates frequently, it costs a huge effort for developers to reconfig the entire system every time manually. Thus, in the automated trigger part, the time parameter was set to automatically trigger the machine learning pipeline every week. The data source updates data every day, with approximately 2,000 new pieces of data. Combined with data timeliness requirements and infrastructure costs, automatically updating the system once a week can not only minimize expenses relative to daily operation, but also make sure that data is updated frequently.

After Extract, Transform, Load (ETL) on the data, the data was fed into the pre-trained machine learning model. In particular, the monitor tool was used in MLOps to supervise the model's performance and check the quality of the data. When the model performance deviates too much from the baseline, it will be discarded decisively.

In this machine learning model, the geographic location entity is a relatively stable data source, especially in modern society, changing the name and attribution of the territory is relatively rare. Therefore, after many discussions, the conclusion was that it is not necessary to continuously train the geolocation entity extraction model. This was the reason why Figure 3.5 has no Continuous Training (CT).

However, the machine learning model was still supervised. When the model performs worse, it can alert developers and use manual work to fix the bad model performance.

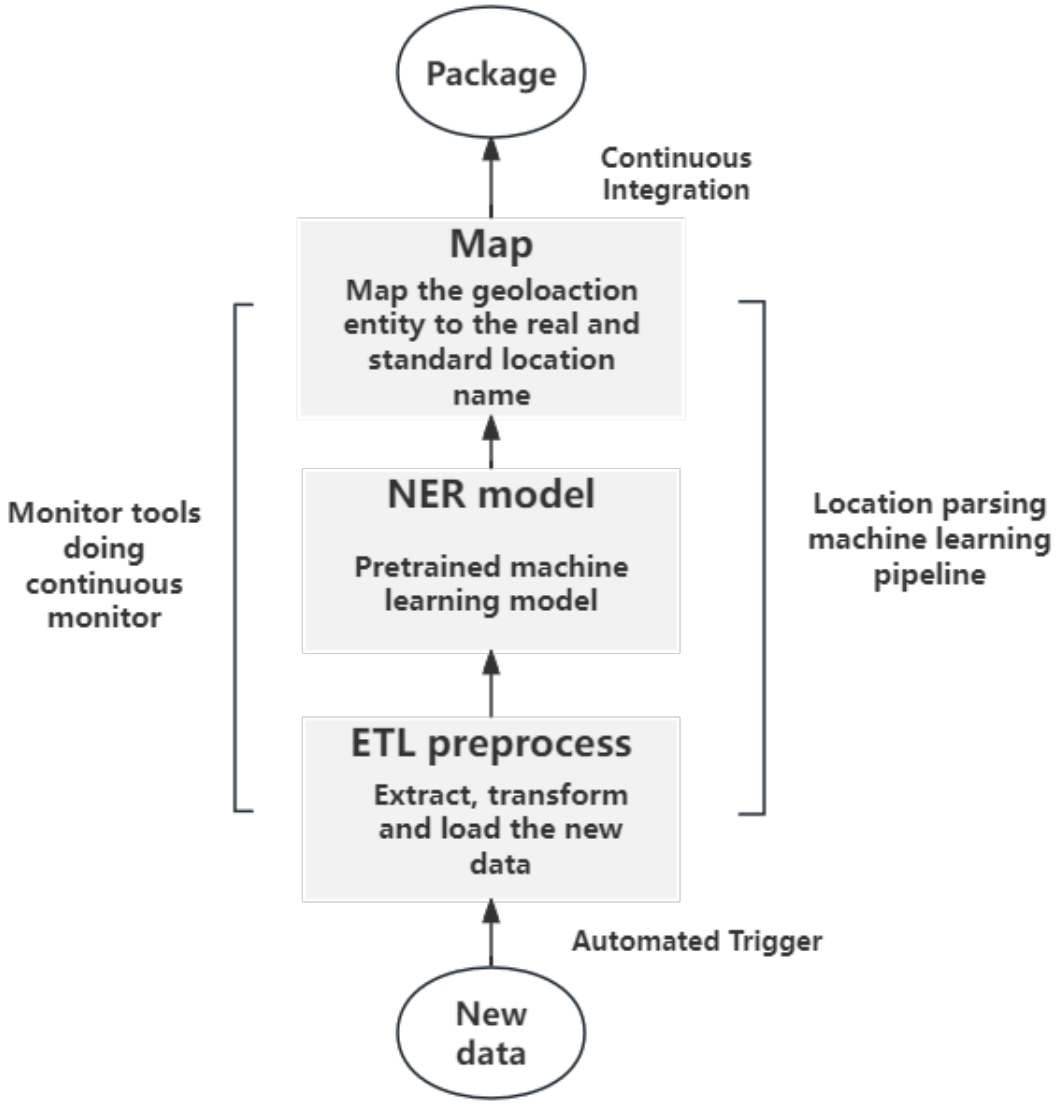
3.2.1.2 Author disambiguation

Similar to the ML automation design for location parsing, a weekly time trigger considered both from data timeliness and infrastructure cost perspective was included in the author disambiguation pipeline to maintain updated data. The complete machine learning automation pipeline is illustrated in the Figure 3.6.

In addition to the time trigger, a continuous training pipeline for author disambiguation was established as the value of the various options changes over time and the model becomes obsolete after several years. In the past, a shared university and field would indicate that they were authored by the same person. However, it is now necessary to include co-authors as a factor to remove the disambiguation.

A monitoring pipeline was established to monitor the performance of the machine learning model. If the newly trained model outperforms the old one, the system will swap the old model to the new one. In this instance, it would aid in maintaining the machine learning model's currency and high level of performance.

Figure 3.5: Location Parsing ML pipeline automation



3.2.2 CI/CD pipeline for the whole system

CI/CD are popular DevOps practices for automation and shortening feedback times [42]. In the system, the CI/CD pipeline is displayed in Figure 3.7.

In the continuous integration pipeline, the latest section of the system is automatically integrated by the system after every successful commit is merged. When it is ready to be deployed, the continuous deployment pipeline would deploy the entire updated system to the appropriate environment, which enables developers to view the results directly.

3.2.3 The entire system MLOps design

In the system, several different data pipelines are constructed. The pipelines can be viewed in the system overview in Figure 3.8. Each red array represents a real pipeline in Jenkins:

- Extracting pipeline and two continuous serving pipelines: They aim to activate the city ranking and author ranking blocks, which include machine learning artifacts for author disambiguation, location parsing artifacts, and other supporting artefacts.
- Continuous deployment pipeline: It deploys the machine learning results to the front-end Server.
- Continuous integration pipeline: It is triggered whenever someone commits the code to GitHub, including unit tests and linting tools.
- Continuous training pipeline: It will automatically trigger the model retraining whenever new training data becomes available.

Figure 3.6: Author disambiguation MLOps pipeline

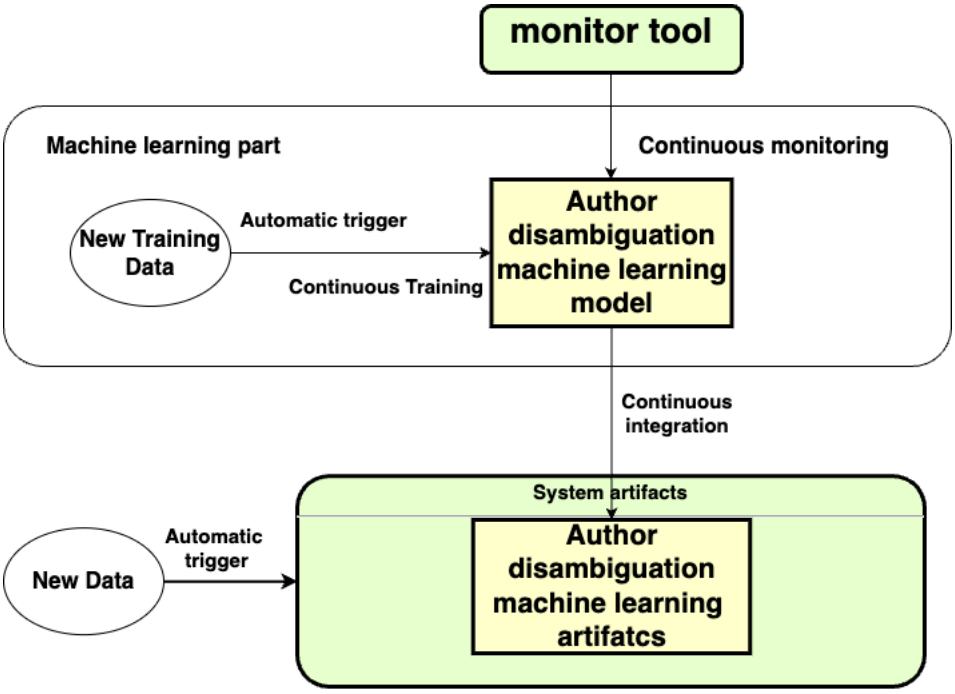
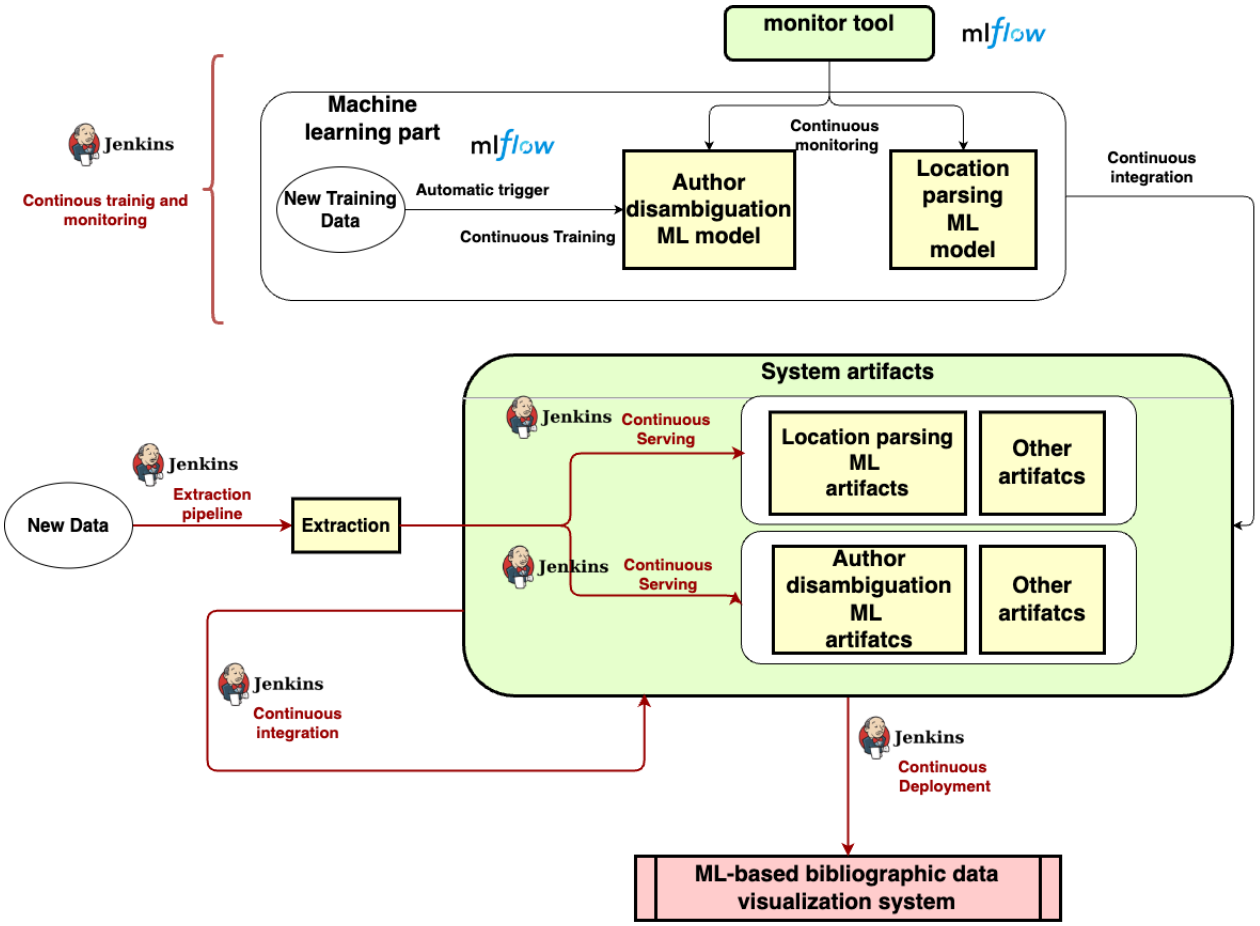


Figure 3.7: CI/CD pipeline

Figure 3.8: System Overview



4

Methods

Both of the research questions lead to in-depth explorations of complex phenomena (machine learning operations) in specific contexts (machine learning based data visualization systems) [43]. This exploration is crucial for evaluating the supremacy of MLOps from three perspectives. The research strategy adopted the methodology of the case study.

A case study methodology, which particularly examines contemporary phenomena in the real-world industry, shapes the research strategy. While case studies may not yield the same type of results as controlled experiments, such as causal correlations, they provide a more thorough comprehension of the phenomenon under research. This makes them well suited for software engineering research where the phenomena of interest, such as MLOps, are complex and not easily isolated for study [44].

The ML based data visualization system is adopted as a case example. To study how MLOps contributes to the development of the visualization system (**RQ 1**), the differences in various aspects before and after using MLOps. As introduced in 3, the practices are derived (**RQ 2**) from the system, which development was completed. In what follows, the research is organized around these two questions. A comprehensive overview of the steps to answer research questions is shown in Figure 4.1 and details will be introduced in the following sections.

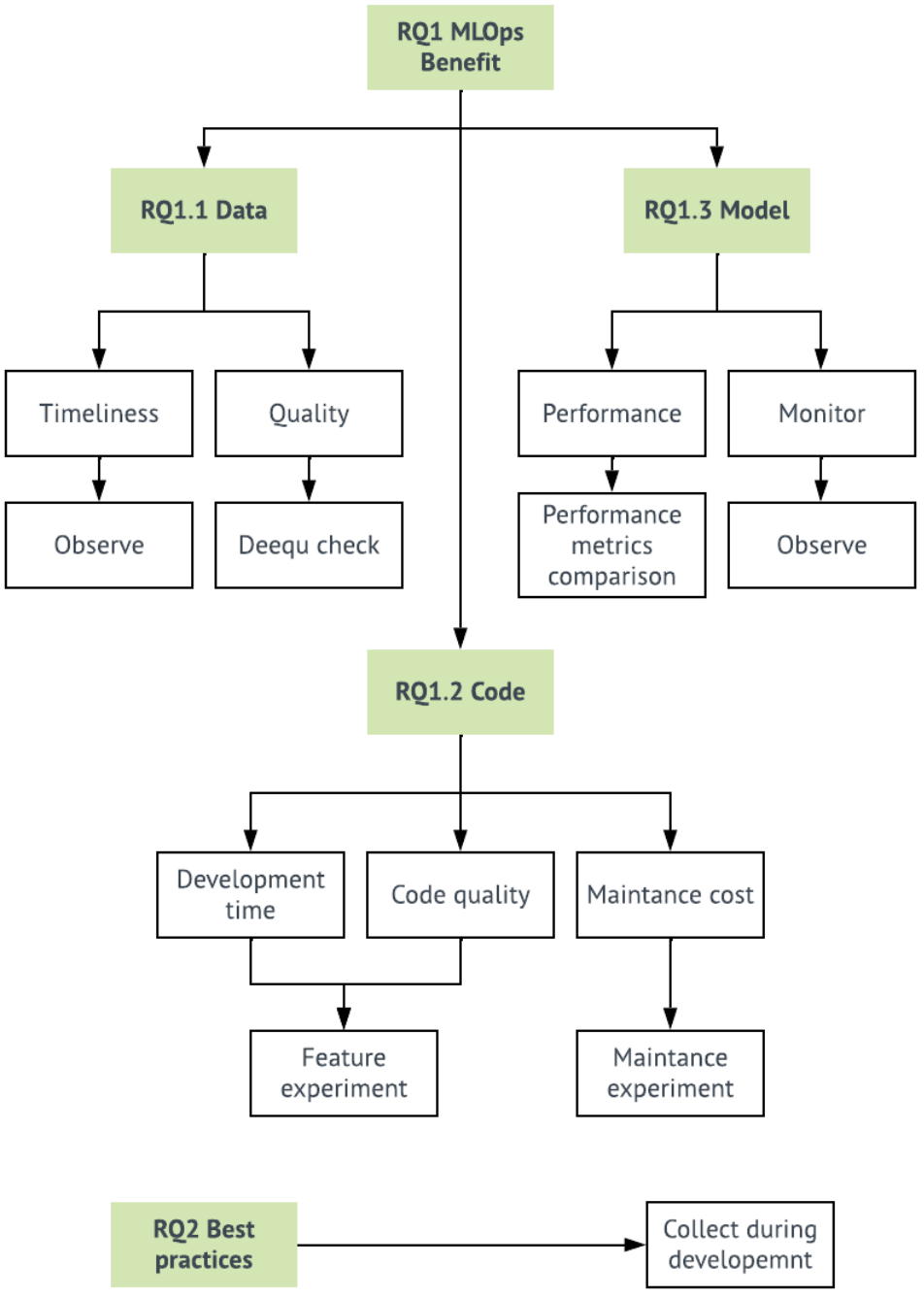
4.1 RQ 1

To evaluate the impact of MLOps, the values of various attributes were compared spanning multiple dimensions across the implementation in the case study. In the multifaceted comparison, the data (**RQ 1.1**), code (**RQ 1.2**), and model (**RQ 1.3**) aspects were evaluated.

4.1.1 Considered quality aspects

The different quality aspects were explored under investigation in the study on MLOps's helpfulness. Unlike previous research in MLOps, which lacks a systematic evaluation framework, the approach draws from both business requirements and standardized quality metrics, i.e., code quality, in the software engineering field.

Figure 4.1: Overview of Case Study Process



This comprehensive evaluation aims to build the gap between theoretical MLOps benefits and practical applications.

A key area where MLOps provides theoretical benefits lies in the management of incoming data. The focus on data management comprised two primary aspects:

- Data timeliness: With a data visualization system, it would be essential to keep the data up-to-date. Compared to current-year data, there is little interest in the outdated 2010 data. Therefore, MLOps' role was determined in maintaining data timeliness in this article.

- Data quality: In terms of data quality, the system addressed two types of input data. One was the training data used to train the machine learning models; the other was the live serving data extracted from the open-sourced online paper website. The quality of the training data is essential for the system because it should meet the standard for training the ML model. Moreover, when the quality of training data is poor, it is not surprising that the model's accuracy is low. Although control was limited over the quality of live serving data sourced from an open-sourced website, the study would assess how MLOps mitigate these challenges.

Furthermore, MLOps is invaluable not only to manage data, but also to enhance the entire software system by integrating the DevOps pipeline. Specifically, in this system, the following aspects were concentrated:

- Development time: Due to the competitive nature of the market, meeting deadlines is crucial for a data visualization project. Thus, if MLOps can help shorten the development cycle, it can optimize additional costs and resources for the system and ultimately increase customer satisfaction.

- Maintenance costs: Maintenance costs generally comprise at least 50% of the project cost [45]. Maintenance costs are a major concern for all research institutions and companies. Almost everyone is eager to minimize these costs to some extent. The authors are curious about the potential of applying MLOps to reduce the project's expense. In theory, automated pipelines, MLOps, can reduce the workload and time required by maintenance developers.

- Code quality: Code quality cannot be overestimated for any project. Since little research supports which MLOps can improve code quality, the authors want to validate this claim for this project.

Finally, MLOps fulfills the system with ML models. The focuses are as follows:

- Model performance: After applying MLOps, many surveys [46, 47] show significant improvement in model accuracy, precision, and other performance metrics. The authors seek to empirically confirm these results in the thesis.

- Monitor: In this aspect, whether MLOps facilitates systems with monitoring

will be discussed.

According to Kobielus' survey conducted in hundreds of enterprises [7], MLOps benefits software development in many aspects. Within the MLOps framework, the experiments are designed to measure improvements in data management, software development efficiency, maintenance costs, code quality, and model performance. As previously mentioned, a real-world example was conducted, and two ML problems were used to demonstrate a MLOps scenario. Furthermore, the authors would provide corresponding practical experience throughout the system development process.

4.1.2 Evaluation

To assess potential improvements in the application of MLOps, evaluation measures were established for the quality attributes stated in Chapter 4.1.1. These measures are tailored to the project's circumstances and the research on different MLOps evaluation papers.

4.1.2.1 Data timeliness

It is challenging to quantify this attribute to see whether the system can be updated on time. Instead, the authors will discuss the extent to which timeliness improves when using MLOps by comparing the system before and after integrating MLOps. In our system's front end, there is a column that shows the most recent data coming date. Therefore, it is straightforward to observe whether the data is the newest or old and the enhancement.

4.1.2.2 Data quality

The assessment of data quality is imperative for ensuring the integrity and reliability of analytical processes. In this study, Deequ was employed, a widely acknowledged data quality testing software developed by AWS Labs and built on Apache Spark. Deequ is specifically designed to formulate "data unit tests" to evaluate the quality of large data sets, making it a suitable choice for our evaluation requirements [48]. Given the system which relies on AWS infrastructure, the use of Deequ ensures seamless integration. Also, it does not require to do any substantial code adaptations to integrate Deequ into the system. Therefore, Deequ was chosen for the evaluation of data quality both before and after the implementation of MLOps with the same tests. The structural results provided by Deequ allowed the authors to justify whether the integration of MLOps had contributed to maintaining high-level data quality standards. In this case, the dataset to test is the same, and except for the MLOps pipeline, nothing changed between the system with and without MLOps, which can make sure little other potential reasons can change the data quality.

There are particular considerations related to data quality that are often evaluated along six dimensions: accuracy, completeness, consistency, uniqueness, timeliness, and validity [49]. In the single-system scenario, there is no necessity to measure

consistency, and it is not easy to assess timeliness and accuracy using current tools. Thus, validity, uniqueness, and completeness are the primary concerns.

The structure of the data inputted into our machine-learning model is outlined as follows:

```
|-- pmid: string (nullable = true)
|-- year: string (nullable = true)
|-- month: string (nullable = true)
|-- day: string (nullable = true)
|-- title: string (nullable = true)
|-- journalTitle: string (nullable = true)
|-- authorList: array (nullable = true)
| |-- element: struct (containsNull = true)
| | |-- lastName: string (nullable = true)
| | |-- foreName: string (nullable = true)
| | |-- initials: string (nullable = true)
| | |-- affiliations: array (nullable = true)
| | | |-- element: string (containsNull = true)
```

In Deequ, it provides a quantifiable measure of the different fields of data quality. Therefore, a comparison between with and without MLOps becomes imperative and gives the authors insights from the quantitative aspect about MLOps helpfulness. Derived from this data structure and the dimensions selected for evaluation, the following assessment plan was formulated:

Completeness: Ensuring the presence of all required data fields with non-null values. Deequ's Completeness analyzer was employed to gauge the percentage of non-null values for each field.

Validity: Enforce the adherence to predefined rules and constraints. The authors focused on fundamental validity principles. Specifically, rules were defined to ensure that "LastName," "ForeName," and "Initials" do not contain numerical characters, while "Year," "Month," "Day," and "Pmid" do not contain alphabetical characters. Deequ's check and custom rules were utilized to enforce these constraints.

Uniqueness: Confirm the absence of duplicate entries in key fields. Acknowledging potential duplications in author names and affiliations due to multiple publications, emphasis was placed on ensuring the uniqueness of "Pmid" as a distinct identifier for each publication. Deequ's uniqueness analyzer was employed to measure the percentage of unique values for "Pmid."

4.1.2.3 Development time

An experiment was conducted to check the development time with MLOps and without MLOps for two specific features. How to setup the experiment and select features and participants is described in Chapter 4.1.3.1. By comparing the time, the authors can determine whether MLOps can help with development time.

4.1.2.4 Maintenance costs

There are various forms of maintenance as per the ISO/IEC 12207 definition, including corrective, preventive, adaptive, and perfective [50]. In the system, adaptive maintenance costs are the main focus, as the system handles a lot of data changes. Maintenance essentially involves adapting the new system with the incoming data. And it would be challenging to monitor all the maintenance work, the authors have decided to simulate the scenario when the system has the new incoming data. An experiment was conducted in this scenario to measure the potential costs associated with these two aspects. Please refer to the next section 4.1.3.2 for a detailed experiment setup.

Maintenance costs arise mainly from two factors: the cost of human resources and infrastructure costs. In terms of human resources, the development time cost in maintenance by developers is calculated based on the time. The authors utilized the average hourly rate for software developers to calculate costs. With regards to infrastructure, the authors evaluated the overall system setup on AWS and compared the AWS bills for system maintenance with and without the use of MLOps. In the system with MLOps, Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3) and Amazon Relational Database Service (Amazon RDS) are employed for the automation pipelines. Amazon EC2 serves as the web server of the system. Amazon S3 and Amazon RDS store the data and machine learning models registry. While in the system without MLOps, only Amazon EC2 is in use to keep the system online. Even though the AWS bill only constitutes a portion of the total infrastructure cost, it serves as an indicator of the potential cost savings of MLOps.

4.1.2.5 Code quality

By encouraging version control, automated testing, and deployment procedures, MLOps is expected to have an impact on code quality by guaranteeing running tests, linting code, and keeping consistency and reproducibility throughout the machine learning development lifecycle.

Many tools on the market, such as SonarQube and Bold, are used to check code quality. Considering that the code repository consists mainly of Python files, Pylint was selected as our tool. It is open source and remains one of the most comprehensive tools on the market, including checking ways of naming, detecting bad practices where the code could be improved, warning of problems specific to Python, probable detection of bugs in the code, and error preventing Pylint execution [51].

An experiment was conducted here to evaluate code quality. During development, the experiment participants tried to develop features in the environment without MLOps and with MLOps, using Pylint to evaluate the quality of the code by detecting code smells. A code smell is a surface symptom that typically points to a more serious issue with the system [52]. Based on the result provided by Pylint, the usefulness of MLOps was judged.

4.1.2.6 Model performance

The implementation of MLOps means continuous monitoring, continuous training, iterative improvement, and systematic deployment of the system, which have a positive impact on model performance.

A classifier model was used in the author disambiguation part of the system described in Chapter 3.1.2. In this thesis,Accuracy, Precision, Recall, and F1-score were chosen to be evaluated, as they are widely used to evaluate the classifier. The change would be observed in the performance matrix before and after applying MLOps to check whether MLOps can help with performance.

4.1.2.7 Model monitor

Compared to a scenario without MLOps, which lacks systematic monitoring processes, the introduction of MLOps allows complete model monitoring through automated tracking, logging, and alerting systems, which ensures fast discovery and mitigation of faults. In this case study, the monitor capability is compared by observation before and after applying MLOps. To what extent the monitoring feature is improved would be discussed when applying MLOps.

4.1.3 Experimental design

As previously mentioned, parts of the evaluation were devised to compare development time, code quality, and maintenance costs with and without MLOps. Experiments could be conducted to study causal relationships, as this is one of the most accurate ways to obtain results. Independent variables, in this case, the use or non-use of MLOps, can be manipulated to enable measurement of their effects on development time and other variables [53].

4.1.3.1 Feature development experiment

Regarding code quality and development time evaluation, the experiment entails that developers at a similar level try to develop two selected features on the system with MLOps and without MLOps. After the experiment, the development time was calculated, and code quality tools were used to obtain the results.

To ensure the accuracy of the experiment's outcome, it is crucial that the features do not affect the independent variables. Thus, no features related to the MLOps pipeline itself should be selected. Additionally, since code quality evaluation would be conducted simultaneously with time evaluation, front end features would not have a significant impact as they typically involve more CSS and HTML code, which is challenging to assess using the chosen tool.

After concrete discussion, the following features were selected:

- **Feature 1**: Update baselines given a certain time frequency.

Set time frequencies are necessary since the data will be consistently uploaded to the corresponding baseline Application Programming Interface (API). To keep the information current, it is imperative to synchronize or update the data from the source website at some intervals. In this case, the file requires refactoring and parameter additions to enable system updates at designated time intervals.

To be more specific, developers should refactor the 'sync_resources' function by adding a new parameter 'time_to_live' under the class 'Register'. The acceptance criteria are that after submission, the code should be able to automatically update the data every week and pass all the existing tests, including the tests which were coded for this feature.

- **Feature 2**: Aggregate all transformed data in the form of JSON from the database.

  To visualize updated data, all transformed data need to be aggregated by cities and authors so that the front-end application can read the data and show the rankings of cities and authors on the website. Regarding the database, in this case, the database was Amazon Simple Storage Service (Amazon S3)

  There was one folder called transformed inside Amazon S3. In this folder, it contained subfolders each of which had one JSON file. Developers need to set it as the structure example and aggregate the transformed data into that standard.

Beyond choosing features, developers are essential to how the experiment is carried out. The developers were carefully selected to have a comparable level of skill with Python, which was the main programming language used in this system, and to have a consistent understanding of the system. Variations in developers' programming abilities or system understanding could introduce significant biases and threaten the validity of the experiment findings.

Based on these criteria, a careful selection procedure was used to choose the two developers for the feature experiment. A survey and screening of individuals known to the authors were conducted, which led to the identification of qualified candidates. Unfortunately, because of the strict requirements on the development skills and background, only two suitable participants are able to join the experiment.

Both developers shared similar academic backgrounds as graduated students from the data science division at Chalmers University of Technology. Additionally, each developer had one year of post-graduation experience working with Python. Their similar proficiency in Python and understanding of this machine learning based data visualization systems established a foundation for a fair comparison.

Given the small number of participants, a specific methodology was employed to improve control over the experimental variables. Each developer was actively engaged in the development of two features. The distinction lied in the use or non-use

of MLOps during these development cycles. Expressly, the first developer was excluded from using MLOps in the first feature development but integrated it to aid the development of the second feature. In contrast, the second developer employed MLOps for the first feature development and excluded its usage for the development of the second feature. Additionally, to avoid that experience with MLOps pipelines would impact their work on features without MLOps, they all worked on the feature without MLOps first.

This methodological approach allows for a more detailed examination of the impact of applying MLOps on feature development. Moreover, the comparative analysis of the overall results obtained from both developers helped mitigate any potential variation in speed and quality caused by individual differences among developers. Therefore, the systematic design of this experiment enables a comprehensive evaluation of the role of MLOps in feature development within this data visualization system.

Regarding the results, assessing code quality is reasonably straightforward since Pylint provides an intuitive evaluation of the code's efficacy. However, accurate tracking timing is crucial, and therefore it is essential to establish a guideline to monitor the time each developer spent on the task. Before the experiment, a comprehensive workshop was held to clarify that developers should only complete the assignment when they have the entire time block.

Other than that, at the beginning, they should commit a "start" message to indicate they begin. When they commit an "end" message, it signifies the completion of the task. A "pause" message denotes a temporary break. Upon resuming, the counting of development time will continue.

4.1.3.2 Maintenance experiment

The maintenance experiment parallels the methodology of the feature experiment, with strict adherence to development time-tracking rules. Unlike the feature experiment, this test does not include an assessment of code quality because developers are only responsible for adaptive maintenance. Additionally, the maintenance task chosen in this experiment is repetitive, allowing the majority tasks to be executed from the command line. This characteristic eliminates the necessity for participants to have an in-depth understanding of the system or to participate in manual code writing. As a result, the difference in developers from various backgrounds is expected to have no influence on the experimental results.

A group of four developers has been carefully chosen to participate in the test. This included two Chalmers master graduates who are currently working as software engineers in the automotive industry, as well as two seasoned software engineers with over five years of experience.

In addition to evaluating developmental time and maintenance outcomes, the costs incurred in AWS would be examined at the conclusion of the experiment.

As previously mentioned, the maintenance tasks chosen involved adaptive changes in the data visualization system. In this experiment, the scenario of data changing was simulated in the system. Developers should update the data visualization system to accommodate upcoming data. The whole process included:

- Extract, transform and load the data to existing machine-learning models.
- Serve the processed data to the front-end visualization system.

The whole process was recorded as the measurement of development time. We first tried this task by ourselves and could see how the city and author rankings changed with the new data. Therefore, the result is easy to evaluate from the front-end rankings.

It was imperative to note that all adaptive changes are fully automated within the MLOps framework because most of the tasks could be run in the terminal with little manual intervention. The reason for favoring adaptive tasks over debugging fixes was the challenge of replicating the debugging environment. Besides, debugging tasks demanded a relatively deep understanding of the system, making it difficult to identify suitable engineers as experiment participants.

Furthermore, in the context of the system, the most common maintenance activity aligned with the chosen experiment. This preference came from the temporal sensitivity of the data visualization system, where the bibliographic information was periodically updated. On the other hand, debugging fixing and other maintenance tasks were occasionally triggered. The strategic alignment of the experiment with the significant maintenance task ensured a more comprehensive and practical evaluation which reflected the system's daily operating requirements.

4.2 RQ 2

For the second research question, the thesis focused on the integration of various MLOps tools and proposed best practices, conducting an exploration of different tools within the MLOps domain during development.

Also, the methodology would be introduced to map state-of-the-art MLOps pipelines identified in the literature mentioned in the background Chapter to the development practices employed in the system during the development. In this case, we adopt the third level MLOps and made CI/CD pipelines and ML pipelines automation.

Practical recommendations are planned to be presented to develop MLOps applications to address these gaps. Particularly, in practice, MLOps adoption is still in its initial stages, some companies are struggling to automate the basic system CI/CD pipelines [54]. Although these lessons would be derived from the experience in data visualization system development, it was aimed at generalizing assumptions. Having explicit assumptions allows other researchers to build more systematic knowledge of best practices from the findings and practitioners to check whether the recommen-

dations might apply to their own application contexts. It would also be discussed to which extent the assumptions might apply to other contexts.

4.3 Summary

This chapter outlined the methodology adopted to address **RQ 1 and RQ 2**. It covered the establishment of quality aspects to different systems with and without MLOps and the evaluation of these aspects. In addition, the assessment included a description of the experiments.

5

Results

After conducting both experimental and observational experiments, the research yielded results. The findings of RQ1 would be presented and analyzed in a sequential manner, following the attributes detailed in Chapter 4.1. Then, the best practices would be discussed through the development process, which answered RQ2.

5.1 Data

As an integral component of a machine learning based data visualization system, data plays a vital role. Among all the attributes related to data, the attributes mentioned in Chapter 4 would be discussed: data timeliness and data quality. These results could be concluded by comparing the use and non-use of MLOps to the system.

5.1.1 Data timeliness

The investigation regarding data timeliness within the case study, the machine learning based data visualization system, revealed improvements facilitated by MLOps.

The system processed bibliographic data from a public platform for this case study to generate city and author rankings. Because of the continuous influx of new papers, this bibliographic data is regularly updated on the open-source website.

Before adopting MLOps, engineers manually executed backend API modifications and front-end adjustments to accommodate updated timeframes. The manual process during the development is a tedious and time-consuming approach, including manually downloading the data, preprocessing the data and then serving the data to the machine learning models, integrating the result into the front-end.

However, with the implementation of MLOps, automated pipelines simplified this update process. Developers set time triggers or data thresholds to enable automatic refreshes without human intervention. These automation pipelines enable a better flow of code from the development to the deployment and the building of better production systems [30]. Additionally, they can decrease the time to market which is fundamental to competitive advantage [55]. Similar to DevOps, the introduction of

Instance	Before Value	After Value
pmid	100/%	100/%
authorList.lastName	99.99/%	100/%
authorList.foreName	99.93/%	100/%
authorList.initials	99.92/%	100/%
title	99.39/%	100/%
jounalTitle	100/%	100/%
year	32.18/%	100/%
month	32.18/%	100/%
day	32.18/%	100/%

Table 5.1: Data Quality Completeness Analysis Result

CI/CD pipelines also reduced the need for software developer involvement, resulting in time savings for the developers and the company [56].

The research demonstrated that integrating MLOps within a machine learning based data visualization system tremendously enhanced data timeliness, ensuring up-to-date information without requiring manual interventions.

5.1.2 Data quality

In this section, the results of the data quality evaluation are presented with Deequ, focusing on the dimensions of completeness, validity, and uniqueness.

1. Completeness

Completeness is a crucial facet of data quality, guaranteeing the inclusion of all essential information. With Deequ, the integrity of the dataset was examined before and after the implementation of MLOps. The results shown in table 5.1 indicate an improvement in data completeness after applying MLOps, particularly in month/day/year data, whereas only marginal improvement is observed in author information. The underlying reason for this disparity is believed to be the high data quality of author information. Before applying MLOps, they already had almost 100% integrity values, leaving little room for enhancement. However, in areas where data quality performance is suboptimal, such as month/day/year data, MLOps demonstrates its capability to enhance data integrity noticeably. For example, the completion value of the month increases from 32.18% to 100%.

The analyses suggestd that applying MLOps contributes to maintaining data integrity, making sure that all necessary information is available for analysis and model training. This improvement is particularly noteworthy in instances where data quality is initially deficient.

2. Validity

Validity is critical to ensure data adherence to predefined rules and constraints. The

evaluation, utilizing Deequ, focused on verifying data against specified criteria, such as the absence of numerical characters in names.

The results obtained are 100% before and after MLOps integration in all the fields, indicating no discernible difference between use MLOps and non-use MLOps, as the original data quality already ensured the fundamental validity. In the original data, names do not contain any numerical character, and there are no alphabetical characters in the time information.

The conclusion here aligns with the preceding section, emphasizing that when data quality assures 100% validity, the application or non-application of MLOps becomes inconsequential to data quality.

3. Uniqueness

Instance	Before Value	After Value
pmid	99.93/%	100/%

Table 5.2: Data Quality Uniqueness Analysis Result

The uniqueness of data is crucial to prevent duplication and ensure the accuracy of analyses. Through the utilization of Deequ, the uniqueness of the dataset was assessed to eliminate any duplicated entries. After implementing MLOps, the data showed improved uniqueness, meeting the uniqueness requirements. This improvement was not significant because of the excellent quality of the original dataset.

In summary, with regards to MLOps, the improvement has relevance to the original quality of the data. It could be inferred that applying MLOps result in a significant enhancement in datasets with lower initial data quality. The advancement was substantial for datasets already of high quality.

The observed improvement in data quality is because when implementing automated data pipelines, it is convenient to integrate various existing plug-ins or basic quality gates to detect data quality issues. In the presence of subpar data quality, the pipeline can reject the entire dataset or screen out the dataset instead of directly sending it to the machine learning model. As a result, an improvement could be observed in the data quality of the training data.

5.2 Code

This section presents the results of quantitative feature experiments based on the evaluation methods discussed in the previous chapter.

5.2.1 Development time

In the feature experiment, the results were properly recorded through the developers' development time of features. Table 5.3 made from the results of the experiment.

TIME/min	FEATURE 1	FEATURE2
Developer A	8'45(without MLOps)	40'23 (with MLOps)
Developer B	30'13(with MLOps)	10'12(without MLOps)

Table 5.3: Feature Experiment Development Time Result

The results suggest that MLOps can increase development time, which contradicts the hypothesis. The authors tried to ask the participants about the reason behind this result and they gave a reasonable outcome. From their perspectives, it can be time-consuming to check or test numerous static codes in MLOps. In the system pipelines, after each commit, the linter, Pylint, accesses the changed files and does the static code analysis. The overall analysis process takes 1-2 minutes on Jenkins, which slightly increases the development time. Additionally, developers have to fix the code smells, and this work costs time as well.

Additionally, these checks often reveal bad coding practices, for example, the absence of public docstrings for public methods. In the code quality part (Chapter 5.2.3), some examples of code smell caught in the feature experiment are shown. Using MLOps increases development time but it prevents lousy code smells from affecting the entire system in the long run. Reluctance to build pipelines can lead to technical debt that far outweighs the extra dozen time spent on development and implementation. It is vital to prioritize long-term development goals over short-term time constraints.

It's important to note that these findings, derived from a small participant pool, might not be universally applicable. The impact of MLOps on development time could vary across different organizational contexts, team sizes, and levels of familiarity with MLOps practices. We anticipate that as teams gain more experience with MLOps, the initial learning curve could flatten, potentially reducing the time overhead associated with these practices.

5.2.2 Maintenance cost

When assessing maintenance costs, it is crucial to consider the cost of human resources and infrastructure. By combining these two factors, a more accurate estimate can be obtained. Maintenance experiments were conducted to evaluate the cost of human resources. The development time of the maintenance task was recorded manually by the authors. Table 5.4 summarizes the experiment's results.

The results indicate that although some developers can complete maintenance tasks quickly, they are still not as efficient as the MLOps pipeline. Given the hourly salary of developers in Sweden, which averages 227.47 Kr/hour according to the statistics of payment company Payscale [57], the use of MLOps pipelines for maintenance work results in significant cost savings.

In addition, the monthly cost of infrastructure bills were also evaluated. In this case, the bill of AWS was collected and compared.

TIME/minutes and seconds	MAINTENANCE TASK
Developer A	15'01
Developer B	13'47
Developer C	24'49
Developer D	15'52
MLOps pipeline	10'23

Table 5.4: Maintenance Time Result

Cost/dollar	AWS EC2	AWS S3	AWS RDS	Overall
Without MLOps	34.104	0	0	34.104
With MLOps	34.104	1.15	0	35.254

Table 5.5: Maintenance AWS cost Result

Table 5.5 shows a slight increase in cost, accounting for 3.26%, after implementing the MLOps pipeline for maintenance tasks. The increase in costs was primarily attributed to the use of Amazon Simple Storage Service (Amazon S3) as the database, which increased the cost of AWS. This impact was relatively small. It was important to note that Amazon Relational Database Service (Amazon RDS) should also be included in the calculation of maintenance costs as part of CT. Thanks to AWS's sales strategy, Amazon RDS could be used for free within a certain range after purchasing Amazon Elastic Compute Cloud (Amazon EC2). Therefore, using Amazon RDS did not significantly impact maintenance costs.

In conclusion, in terms of infrastructure costs, using a MLOps pipeline generally increased costs by $1.15 per month, 3.26%.

To provide a holistic view, two tables (5.4 and 5.5) would be merged. Here, it is supposed that developers would do the maintenance task once every month. Figure 5.1, based on the current US dollar (1 SEK = 0.097 USD) exchange rate, visually demonstrates the total cost consumption.

From Figure 5.1, it is evident that employing MLOps significantly decreases maintenance costs, substantiating the cost-effectiveness of MLOps pipelines in software development.

5.2.3 Code quality

The feature experiment also sought to systematically assess and compared the code quality in environments with and without MLOps integration. Pylint, a widely employed code analysis tool, was utilized to evaluate the quality of the codebase. A standard Pylint output is a simple terminal output that contains all of the warnings, error messages, convention problems, and fundamental code statistics. At the conclusion of the output, a total rating score is given as a general indicator of the code quality. The rating score is determined by Pylint using the formula below, where

Figure 5.1: Total Cost Consumption Comparison

"statement" is the total number of lines of code that Pylint has examined [58].

$$\text{Pylint Score} = 10.0 - \left(\frac{5 \times \text{error} + \text{warning} + \text{refactor} + \text{convention}}{\text{statement}} \right) \times 10$$

To establish a baseline, Pylint scores were initially recorded as 10.0/10.0 which means the original code does not have any code smells, providing a reference point to identify potential issues that may arise during subsequent feature development.

The comparative analysis of average Pylint scores, detailed in Table 5.6, was conducted across various features and developers. The table captures the nuanced variations in Pylint scores, explicitly focusing on the impact of MLOps integration of code quality.

Pylint Score	**Feature 1**	**Feature 2**
Developer A	9.84/10 (Without MLOps)	10/10 (With MLOps)
Developer B	10/10 (With MLOps)	9.7/10 (Without MLOps)

Table 5.6: Pylint Scores in Feature Experiment

Table 5.6 reveals intriguing insights into the code quality variations among develop-

ers and features operating within distinct development environments. Both developers A and B exhibited higher Pylint scores in the MLOps environment, indicating a correlation between MLOps integration and enhanced code quality, even when dealing with different features.

Furthermore, the comprehensive Pylint output in the non-MLOps environment illustrated the specific issues identified during the code quality analysis. The code was rated at 9.84/10, with notable concerns highlighted. The score looks quite high, but considering the industry standard and the large number of files in the repository, a missing 0.1 score can indicate over 10 different code smells, such as undefined variables, suboptimal import order, and warnings about unused variables. These may introduce some troubles in future software development.

```
************* Module backend.aggregate.aggregate
backend/aggregate/aggregate.py:19:57: E0602: Undefined variable 'inp
ut_dir' (undefined-variable)
backend/aggregate/aggregate.py:20:42: E0602: Undefined variable 'gl
ob' (undefined-variable)
backend/aggregate/aggregate.py:20:55: E0602: Undefined variable 'inp
ut_dir' (undefined-variable)
backend/aggregate/aggregate.py:37:9: W1514: Using open without expli
citly specifying an encoding (unspecified-encoding)
backend/aggregate/aggregate.py:25:12: W0612: Unused variable 'addre
ss'(unused-variable)
backend/aggregate/aggregate.py:6:0: C0411: standard import "import
json"should be placed before
            "from backend.lib.s3_client_handler import AwsS3Client"
            (wrong-import-order)
backend/aggregate/aggregate.py:7:0: C0411: standard import "from pa
thlib import Path" should be placed before
            "from backend.lib.s3_client_handler import AwsS3Client"
            (wrong-import-order)
backend/aggregate/aggregate.py:7:0: W0611: Unused Path imported from
pathlib (unused-import)
backend/aggregate/aggregate.py:14:0: I0021: Useless suppression of
'unused-argument' (useless-suppression)

-------------------------------------------------------------------
Your code has been rated at 9.84/10 (previous run: 10.00/10, -0.16)
```

The observations derived from the feature experiment align with the hypothesis and existing literature, underscoring the positive influence of MLOps on software development processes [51]. The MLOps-integrated environment showcased notable improvements in code quality, emphasizing the efficacy of automated procedures facilitated by MLOps in maintaining coding standards, identifying issues, and ultimately elevating the overall quality of the codebase.

5.3 Models

In this section, the evaluation results which conducted on the model aspects of the system are presented, focusing on the accuracy and the monitor feature.

5.3.1 Performance

Regarding accuracy, as discussed in Chapter 4.1.2, a continuous training pipeline was not established for the location parsing model. Therefore, a direct performance comparison before and after MLOps implementation is not feasible for this component. The result of evaluating the author disambiguation classifier model would be discussed only.

The author disambiguation component of the system, as detailed in Chapter 3.1.2, employed a classifier model. The model performance was evaluated on standard metrics, including Accuracy, Precision, Recall, and F1-score. These metrics are commonly utilized in assessing classifier performance.

Without MLOps, the authors manually run the training process when new training data is available. The training process 0-6 times was triggered manually, resulting in the performance metrics shown in Figure 5.2. Subsequently, Figure 5.3 presents the outcomes of six executions of the continuous training pipeline following the implementation of MLOps. Notably, in both figures, the result at 0 times represents the original outcome before any MLOps-driven interventions.

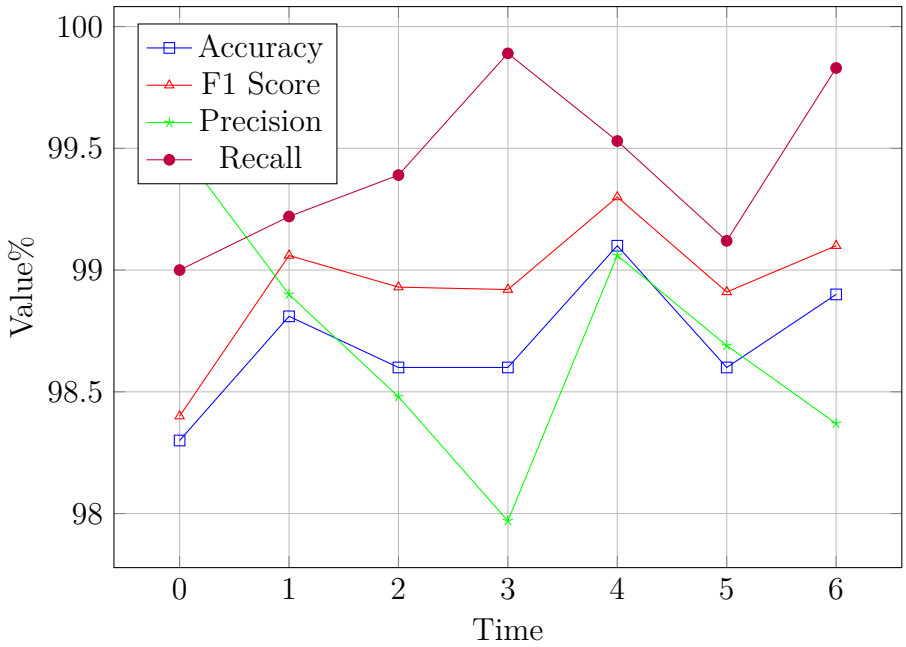


Figure 5.2: Machine Learning Metrics without MLOps Over Time

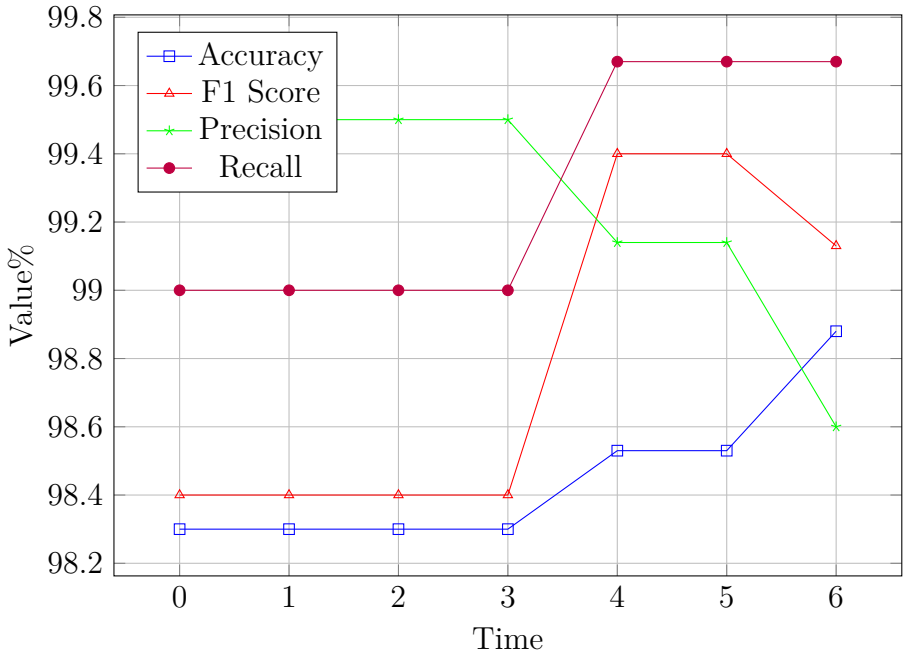


Figure 5.3: Machine Learning Metrics with MLOps Over Time

When developers manually trained the model multiple times, it is evident that the accuracy value exhibited a similar increasing trend but volatile results compared to the results obtained using MLOps. This result is because in the MLOps environment, a data check plug-in is added, and the data sent in the MLOps is of higher quality. In addition, in the MLOps environment, a filter for machine learning model is set, if a lower quality model is received, the model would be dropped immediately instead of integrated into the system. In the context without applying MLOps, the accuracy reaches a high value of about 99% in the 4th iteration, and then it decreases rapidly to 98.6% in the fifth iteration. The recall value also goes down from approximately 99.7% to 99.1% between the 3rd and 5th iteration. As a comparison, the accuracy has a steady increment in the iterations with MLOps integration from 98.3% to 98.9%, as well as the Recall value from 99% to 99.7%.

The reason is that MLOps offers automated, systematic, and continuous training processes, contrasting with the manual triggering processes in a local environment. Manually triggering the process may lack automation, potentially leading to unpredictable performance metrics changes. For example, model accuracy decreases rapidly which can be found in the 5th iteration in Figure 5.2. MLOps provides a structured framework to ensure continuous integration and deployment, reducing manual intervention, and enhancing reproducibility.

Furthermore, in Figure 5.2 and Figure 5.3, F1 score, Precision, and Recall metrics display noticeable fluctuations. The values keep up and down several times in Figure 5.2. The Precision value even falls to 98% in the 3rd iteration. However, in the MLOps environment, they remain more stable. Besides, this observed instability in MLOps is due to the filtering criteria which were set during continuous training.

These criteria do not highly consider F1, precision, and Recall metrics, which introduce a slightly unstable factor in the MLOps environment compared to the steady increment of accuracy.

Overall, in the same scenario where the pipeline will be triggered manually or automatically when training data is received, MLOps can keep the model performance increasing in a more stable way instead of introducing fluctuations in the system, which cause prediction results to be unsteady on the user end. Additionally, this comparison was set up in a situation where the developer manually triggered the pipeline whenever new training data was available. In practice, developers sometimes find it difficult to recall this. It is quite likely that developers forget to update, leading to the obsolescence of the machine learning model. Instead, MLOps enables the model to adjust to changing data patterns without the need for manual intervention.

In this evaluation, the author disambiguation classifier demonstrates a steady enhancement in performance metrics after applying MLOps, underscoring the positive impact of MLOps integration on machine learning performance.

5.3.2 Monitor

Upon the application of MLOps to the system, a notable enhancement was observed in the monitoring capabilities of machine learning models. Preceding the adoption of MLOps, it was possible to manually log metrics such as accuracy and performance after each training session through code. However, with the integration of MLOps, a substantial portion of these metrics was automatically stored locally or in specified cloud storage.

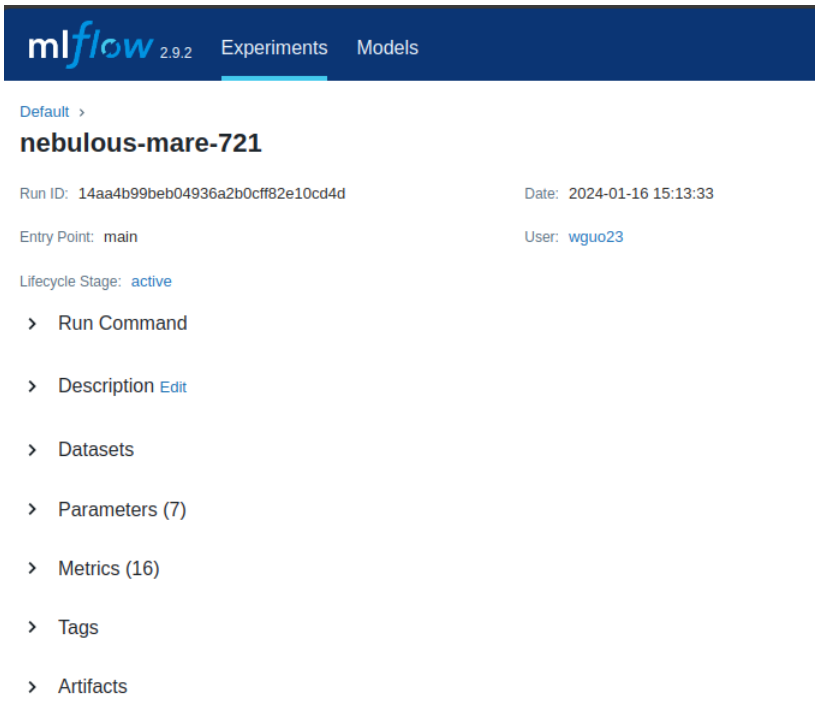
For instance, in the case study presented, the integrated monitoring feature within ML flow provides comprehensive monitoring functionalities. As illustrated in Figure 5.4, it not only captures various performance metrics of the model but also automatically preserves training parameters and datasets and generates artifacts based on the data. Furthermore, ML flow stores each model iteration after training. In the event of unforeseen issues with a newly deployed model version, quick rollbacks to the previous version can be executed, ensuring system stability.

In conclusion, the application of MLOps can be asserted to significantly elevate the monitoring capabilities of machine learning models within the system. This augmentation subsequently fortifies the system's ability to detect anomalous behaviours, ensuring robustness and stability.

5.4 Best practises

To address the **RQ 2**, this study systematically evaluated the effectiveness of the integrated MLOps practices in developing a machine learning based data visualization system. The investigation focused on the combined usage of various MLOps tools,

Figure 5.4: ML pipeline Monitor



aiming to identify best practices. Additionally, it bridges existing gaps in deploying and maintaining machine learning models. The research methodology involved exploring different tools in the MLOps field. It also included mapping state-of-the-art MLOps practices, which were identified in the literature, to the specific development practices employed in the system.

Tool Selection Based on System Requirements: There are various MLOps tools on the market to satisfy the need for MLOps. A selection of these tools were experimented with, such as Airflow and ML flow [11], during developing the system. In this case study, because data process is straightforward in the system, ML flow is chosen to be integrated rather than Airflow. While Airflow offers comprehensive coverage across the machine-learning lifecycle, the learning and setup are tough. ML flow specializes in aspects critical to machine-learning workflows and is easier to integrate. This finding shows the importance of selecting MLOps tools that align with the project's specific needs.

Integrating MLOps from the Start: MLOps should be developed in conjunction with the overall system to avoid compatibility difficulties. In this case, the system was created first, and then all MLOps pipelines were added during the development process, which was driven by experimental goals. This considerably increased the developmental challenges later, as many tool interfaces did not match the early development preferences. Furthermore, if the pipelines can be integrated from the very beginning, developers would have more experience on the MLOps pipelines. This might help with limiting the increment in the development time in the MLOps

environment.

Leveraging Cloud Computing for Efficiency: The adoption of cloud computing emerged as a significant facilitator in this study, offering scalability and resource optimization not available in local setups. Particularly, cloud environments allow for dynamic resource allocation and enhancing sustainability. As the result of the maintenance experiment shows, even though MLOps consumed a lot more resources, the infrastructure cost did not go up much higher. Additionally, current MLOps tools offer extensive support on major cloud platforms, it is very easy to set up from our experience.

In conclusion, the findings of this study provide important insights into the efficacy of integrated MLOps techniques. By carefully selecting tools based on project needs, integrating MLOps practices from the outset, and leveraging cloud computing, organizations have the possibility to harness the benefits of MLOps—improving code quality and system reliability—without incurring significant increases in development time and infrastructure cost. These recommendations are grounded in the specific findings of this research and are intended to provide actionable guidance for practitioners seeking to optimize their development efforts with MLOps.

5.5 Discussion

5.5.1 Comparison to previous findings

After analyzing the results, a brief comparison was conducted between the findings and the existing studies.

By examining the impact of MLOps on the data aspect, it was found that automating data processing pipelines significantly improves data timeliness, an area that was previously not addressed by existing research. This improvement in timeliness is attributed to the transparency of data visualization. Consistent with the prior findings of [5, 59, 60], this study confirms the critical weight of data quality checks within the MLOps process, leading to improved data quality. Unlike in the study [5], the degree of improvement was lower due to the high quality of the original dataset and the use of different data processing tool in this MLOps pipeline.

In terms of code quality, similar to previous findings [17, 61], this research highlights the inherent DevOps benefits within MLOps technique, including improved code quality and reduced maintenance cost. Contrary to the opinion of [62, 63, 64] that linked CI/CD can shorten feature delivery time, the authors observed an increase in development time. Combining with the maintenance cost and code quality result to see, the increment in development time is a trade-off for higher coder quality and easier further maintenance. Software quality and following code standards can not be underestimated in software development. Neglecting these aspects negative consequences in the future [65].

The analysis of model evaluation reinforces MLOps positive impact on model qualify

[46, 47], shown by the stable increase in performance. The monitoring functionality was also improved by applying MLOps, with minimal discussion of the findings.

Nonetheless, it is crucial not to underestimate the discrepancies between the results and previous research. For example, some studies [62, 63, 64] believe that applying MLOps can reduce development time, but this research showed a subtle relationship between MLOps and development productivity. Moreover, the results did not indicate a significant enhancement in the performance indicators. In the case study, MLOps could only make the increment steady and prevent fluctuations. This disparity demonstrates that the multifaced nature of MLOps implementation and different background factors can shape the outcome in some way.

In summary, the findings are consistent with many established findings in the literature in the MLOps field, and the results also provided new insights and nuances to the discussion.

5.5.2 Threats to validity

The research aims to provide valuable insights into the application of MLOps in the development of a machine learning based data visualization system, it is essential to recognize potential threats to the validity of the findings. Validity refers to the relationship between experimental conclusions and reality and how the findings might not be correct [66]. Wohlin proposes four types of validity threats: conclusion, internal, construct, and external [67]. The validity of threats based on these four types will be discussed:

5.5.2.1 Internal validity

There are factors that could impact the assessment of the causal relationship between the implementation of MLOps and observed improvements in data quality, code quality, and model quality. For instance, the model type which was implemented in the system. Unaccounted factors could have an impact on the results and challenge the internal validity of the research.

5.5.2.2 External validity

This study's exploration of MLOps practices within the data visualization field primarily took place within the context of a single organization, Software by Quokka. This unique setting offers insights into the practical implementation but also necessitates caution when generalizing the findings to other organizational contexts. For instance, Software by Quokka's resource allocation and specific data visualization needs could influence the effectiveness and applicability of certain MLOps practices observed in this study.

Additionally, the study primarily investigates the application of MLOps in the data visualization field, acknowledging potential limitations in generalizing the findings to other domains due to the unique requirements inherent in different domains, which could affect the effectiveness of MLOps practices. Additionally, the utilization of

specific MLOps tools like ML flow [11] and Deequ [48] limit the applicability of the best practices to users with other tools, as differences in tool functionalities and interfaces could influence the external validity of our recommendations.

5.5.2.3 Construct validity

Construct validity refers to the degree to which conclusions about the theoretical constructs that support the operationalization in the study can be drawn with reasonableness [68].

The operationalization of concepts in this case study, such as data quality, code quality, and model quality, involves specific metrics and tools. The selection of these metrics and tools may introduce biases or limitations in the construct validity of the measurements.

Furthermore, using standard metrics (Accuracy, Precision, Recall, F1-score) to evaluate the machine learning model may not fully capture the difference in model performance. It may lead to the threat to the construct validity of model quality assessment.

5.5.2.4 Statistical conclusion validity

Statistical conclusion calidity pertains to how much data from a research study can be considered to legitimately be regarded as demonstrating a link (or lack thereof) between independent and dependent variables [69].

The size and representativeness of the experiments were quite small. In the feature experiment, there were two participants. And they only worked for one year. In the maintenance experiment, there were four participants. This could impact the statistical power of the results. Also, the unique properties of the dataset employed in the investigation could have an impact on the generalizability of statistical conclusions.

Additionally, the statistical analyses conducted, such as analyses for development time and maintenance costs, were subject to potential Type I or Type II errors. Variability in the data and assumptions underlying statistical tests introduce threats to the conclusion validity of the quantitative results.

5.5.2.5 Use of ChatGPT

In addition to the above four types of threats, with the increasing use of ChatGPT [70], this thesis introduced another threat to validity: the use of ChatGPT. Section 4.1.2.2 and Section 4.1.3 under Chapter 4, most sections under Chapter 5, and Chapter 6 adopted ChatGPT to suggest grammar corrections and rephrasing. This could introduce some phrasing and language biases that the authors cannot detect.

5.6 Summary

The Results chapter systematically presented the outcomes of experimental and observational work across various dimensions of a machine-learning-based data visualization system. The investigation encompassed data aspects, code development, maintenance, code quality, and model evaluations. The findings included substantial improvements in data timeliness through MLOps integration, enhancements in data quality, cost savings in development and maintenance tasks, improved code quality with MLOps, and performance improvements in the author disambiguation classifier model. Furthermore, the monitoring capabilities of machine learning models were significantly elevated through MLOps, ensuring robustness and stability.

Furthermore, addressing RQ 2, the study provided best practices in the integrated use of MLOps tools. It highlighted the importance of aligning MLOps development with the entire system to mitigate compatibility issues. Additionally, concurrent development of MLOps and system components was recommended. It also emphasized the seamless integration of cloud computing to enhance efficiency and sustainability. The study also compared the functionalities of MLOps tools, suggesting Airflow for comprehensive system coverage and ML flow for a more specialized focus on machine learning workflows.

6

Conclusion

To address the **RQ 1** in Chapter 1.2, "To what extent can MLOps be helpful during the engineering of a data visualization system?" and its sub-questions, the study provided comprehensive insights into the multifaceted impacts of MLOps on various aspects of system development and maintenance.

Data Quality Enhancement (RQ1.1): The findings certainly indicated the importance of MLOps in improving data quality in a machine learning based data visualization system. MLOps automates data updating operations, decreasing manual intervention, minimizing errors, and ensuring up-to-date information. The systematic evaluation using Deequ [48] further supports the assertion that MLOps contributes to data completeness and validity.

Code Quality Improvement (RQ1.2): The application of MLOps proved instrumental in improving code quality throughout the development process. By facilitating automated checks and tests, MLOps identifies and rectifies coding issues, ensuring adherence to coding standards. The feature experiment results, in terms of development time and maintenance costs, provided tangible evidence which the slight increase in development time is outweighed by the long-term benefits of code quality assurance through MLOps.

Model Quality Advancements (RQ1.3): While the continuous training pipeline limitations constrained a direct comparison of model performance, the evaluation of the author disambiguation classifier model showed improvements in stable increment of accuracy after MLOps implementation. Additionally, the monitoring capabilities of machine learning models were notably enhanced through MLOps, providing automated metrics tracking, model versioning, and rapid quick rollbacks for system stability.

To answer the other question, "Which best practices can be derived from applying MLOps to such a visualization system?" (**RQ2**), this study explored the unknown realm of MLOps applications in the data visualization field. Recognizing the absence of established practices in this domain, the study served not only as an empirical examination, but also as a pioneering attempt to provide a realistic example for future practitioners, developers, and researchers.

Overall, our findings not only gave empirical evidence of MLOps' effectiveness in data visualization but also defined core best practices for future attempts in similar areas. By exploring uncharted territory, the research contributed to the development of norms, establishing a shared understanding of effective MLOps applications. As practitioners, developers, and academics start on new initiatives at the convergence of MLOps and data visualization, the findings from the research can provide a guiding light for further improvements.

In the future, studies could dive deeper into evaluating model performance metrics beyond standard measures such as Accuracy, Precision, Recall, and F1-score. Also, this research primarily focused on the application of MLOps in the context of data visualization, future studies could extend this exploration to other domains. Investigating how MLOps practices vary across different domains and assessing their generalizability could provide valuable insights for practitioners in diverse fields. Furthermore, conducting longitudinal studies to track the long-term effects of MLOps integration could offer valuable insights into the scalability and evolution over time.

Bibliography

[1] Microsoft Corporation. (n.d.). "Bing." Accessed: Tuesday 4th June, 2024, [Online]. Available: https://www.bing.com/.

[2] Apple Inc. (n.d.). "Siri." Accessed: Tuesday 4th June, 2024, [Online]. Available: https://www.apple.com/siri/.

[3] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and 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.

[4] W. Contributors, *Devops*, Dec. 2019. [Online]. Available: https://en.wikipedia.org/wiki/DevOps.

[5] C. Renggli, L. Rimanic, N. M. Gürel, B. Karlaš, W. Wu, and C. Zhang, "A data quality-driven view of mlops," *arXiv preprint arXiv:2102.07750*, 2021.

[6] B. Akinremi, *Mlops tools for nlp projects*, Jul. 2022. [Online]. Available: https://neptune.ai/blog/mlops-tools-for-nlp-projects.

[7] J. Kobielus, *TDWI Best Practices Report | Unifying Data Management and Analytics Pipelines*. 2022.

[8] C. Breuel, *Mlops: Machine learning as an engineering discipline | built in*, Oct. 2022. [Online]. Available: https://builtin.com/machine-learning/mlops.

[9] *Mlops: Continuous delivery and automation pipelines in machine learning | cloud architecture center*, May 2023. [Online]. Available: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning.

[10] N. Hewage and D. Meedeniya, "Machine learning operations: A survey on mlops tool support," 2022. DOI: 10.48550/ARXIV.2202.10169. [Online]. Available: https://arxiv.org/abs/2202.10169.

[11] *Mlflow - a platform for the machine learning lifecycle*. [Online]. Available: https://mlflow.org/.

[12] *Apache airflow*, 2023. [Online]. Available: https://airflow.apache.org/.

[13] V. Anh, *Airflow, mlflow or kubeflow for mlops?* Mar. 2022. [Online]. Available: https://aicurious.io/blog/2022-03-26-airflow-mlflow-or-kubeflow-for-mlops.

[14] *Kubeflow*, 2023. [Online]. Available: https://www.kubeflow.org/.

[15] *Neptune| experiment tracking tool for you and your team*, 2023. [Online]. Available: https://neptune.ai/.

[16] S. Alla, *Beginning MLOps with MLFlow: Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure*. Apress Berkeley, CA, 2021, ISBN: 9781484265482.

[17] DataRobot, *Datarobot acquires mlops pioneer and category leader, parallelm*, Jun. 2019. [Online]. Available: https://www.datarobot.com/newsroom/press/datarobot-acquires-mlops-pioneer-and-category-leader-parallelm/.

[18] Y. Luo, X. Qin, N. Tang, and G. Li, "Deepeye: Towards automatic data visualization," Apr. 2018, pp. 101–112. DOI: 10.1109/ICDE.2018.00019.

[19] Y. Wang, Z. Jin, Q. Wang, W. Cui, T. Ma, and H. Qu, *Deepdrawing: A deep learning approach to graph drawing*, 2019. arXiv: 1907.11040 [cs.HC].

[20] C. Chen, C. Wang, X. Bai, P. Zhang, and C. Li, "Generativemap: Visualization and exploration of dynamic density maps via generative learning model," *IEEE Transactions on Visualization & Computer Graphics*, vol. 26, no. 01, pp. 216–226, Jan. 2020, ISSN: 1941-0506. DOI: 10.1109/TVCG.2019.2934806.

[21] O. Uncu, W. A. Gruver, D. B. Kotak, D. Sabaz, Z. Alibhai, and C. Ng, "Gridbscan: Grid density-based spatial clustering of applications with noise," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 2006, pp. 2976–2981. DOI: 10.1109/ICSMC.2006.384571.

[22] K.-S. Srl, *Bibliometrix - home*, 2023. [Online]. Available: https://www.bibliometrix.org/home/.

[23] S. Mäkinen, H. Skogström, E. Laaksonen, and T. Mikkonen, "Who needs mlops: What data scientists seek to accomplish and how can mlops help?" *CoRR*, vol. abs/2103.08942, 2021. arXiv: 2103.08942. [Online]. Available: https://arxiv.org/abs/2103.08942.

[24] R. Subramanya, S. Sierla, and V. Vyatkin, "From devops to mlops: Overview and application to electricity market forecasting," *Applied Sciences*, vol. 12, no. 19, p. 9851, 2022.

[25] E. Raj, D. Buffoni, M. Westerlund, and K. Ahola, "Edge mlops: An automation framework for aiot applications," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 191–200. DOI: 10.1109/IC2E52221.2021.00034.

[26] J. F. Rodrigues Jr, A. Traina, and C. Traina Jr, "Enhancing data visualization techniques," in *Third IEEE Intl. Workshop on Visual Data Mining-VDM@ ICDM03*, 2003, pp. 97–112.

[27] X. Qin, Y. Luo, N. Tang, and G. Li, "Making data visualization more efficient and effective: A survey," *The VLDB Journal*, vol. 29, no. 1, pp. 93–117, 2020.

[28] R. T. Sutton, D. Pincock, D. C. Baumgart, D. C. Sadowski, R. N. Fedorak, and K. I. Kroeker, "An overview of clinical decision support systems: Benefits, risks, and strategies for success," *NPJ digital medicine*, vol. 3, no. 1, p. 17, 2020.

[29] Q. Wang, Z. Chen, Y. Wang, and H. Qu, "A survey on ml4vis: Applying machine learning advances to data visualization," *IEEE transactions on visualization and computer graphics*, vol. 28, no. 12, pp. 5134–5153, 2021.

[30] M. N. Chowdary, B. Sankeerth, C. K. Chowdary, and M. Gupta, "Accelerating the machine learning model deployment using mlops," *Journal of Physics: Conference Series*, vol. 2327, no. 1, p. 012 027, Aug. 2022. DOI: 10.1088/1742-

6596/2327/1/012027. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2327/1/012027.

[31] N. L. of Medicine, *Pubmed labs*, 2023. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/.

[32] K.-J. Stol and B. Fitzgerald, "The abc of software engineering research," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 3, Sep. 2018, ISSN: 1049-331X. DOI: 10.1145/3241743. [Online]. Available: https://doi.org/10.1145/3241743.

[33] M. Dredze, M. J. Paul, S. Bergsma, and H. Tran, *Carmen: A twitter geolocation system with applications to public health*, 2013. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.309.6126.

[34] *Openeventdata/mordecai_2023*, Jul. 2023. [Online]. Available: https://github.com/openeventdata/mordecai.

[35] *Geoparse 2.0.3*, Feb. 2021. [Online]. Available: https://pypi.org/project/GEOparse/.

[36] *Apache spark™ - unified engine for large-scale data analytics*, 2022. [Online]. Available: https://spark.apache.org/.

[37] V. Kocaman, *Comparison of key medical nlp benchmarksspark nlp vsaws, google cloud and azure*, May 2022. [Online]. Available: https://medium.com/spark-nlp/comparison-of-key-medical-nlp-benchmarks-spark-nlp-vsaws-google-cloud-and-azure-cab5619d2bf6.

[38] *Geonames*. [Online]. Available: https://www.geonames.org/.

[39] Kaggle, *Kaggle: Your home for data science*, 2022. [Online]. Available: https://www.kaggle.com/.

[40] D. K. Sanyal, P. K. Bhowmick, and P. P. Das, "A review of author name disambiguation techniques for the pubmed bibliographic database," *Journal of Information Science*, vol. 47, no. 2, pp. 227–254, 2021. DOI: 10.1177/0165551519888605. eprint: https://doi.org/10.1177/0165551519888605. [Online]. Available: https://doi.org/10.1177/0165551519888605.

[41] K. Jhawar, D. Sanyal, S. Chattopadhyay, P. Bhowmick, and P. Das, "Author name disambiguation in pubmed using ensemble-based classification algorithms," Aug. 2020, pp. 469–470. DOI: 10.1145/3383583.3398568.

[42] *Best practices ebook | perforce*. [Online]. Available: https://www.perforce.com/p/kw/ci-cd-best-practices-for-embedded-software.

[43] Y. Rashid, A. Rashid, M. A. Warraich, S. S. Sabir, and A. Waseem, "Case study method: A step-by-step guide for business researchers," *International Journal of Qualitative Methods*, vol. 18, p. 1 609 406 919 862 424, 2019. DOI: 10.1177/1609406919862424. eprint: https://doi.org/10.1177/1609406919862424. [Online]. Available: https://doi.org/10.1177/1609406919862424.

[44] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[45] C. Orlando, *Software maintenance cost*, Oct. 2023. [Online]. Available: https://galorath.com/software-maintenance-costs/.

[46] Y. Zhou, Y. Yu, and B. Ding, "Towards mlops: A case study of ml pipeline platform," in *2020 International conference on artificial intelligence and computer engineering (ICAICE)*, IEEE, 2020, pp. 494–500.

[47] R. B. Bahaweres, A. Zulfikar, I. Hermadi, A. I. Suroso, and Y. Arkeman, "Docker and kubernetes pipeline for devops software defect prediction with mlops approach," in *2022 2nd International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*, IEEE, 2022, pp. 248–253.

[48] Awslabs, *Awslabs/deequ: Deequ is a library built on top of apache spark for defining "unit tests for data", which measure data quality in large datasets.* [Online]. Available: https://github.com/awslabs/deequ.

[49] S. Gawande, *A guide for data quality (dq) and 6 data quality dimensions*, Jul. 2023. [Online]. Available: https://icedq.com/6-data-quality-dimensions.

[50] "Iso/iec/ieee international standard for software engineering - software life cycle processes - maintenance," *ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998*, pp. 1–58, 2006. DOI: 10.1109/IEEESTD.2006.235774.

[51] Alexandre, *Pylint: How to boost productivity through code analysis*, Jun. 2023. [Online]. Available: https://datascientest.com/en/pylint-how-to-boost-productivity-through-code-analysis.

[52] M. Fowler, *Bliki: Code smell*, Feb. 2006. [Online]. Available: https://martinfowler.com/bliki/CodeSmell.html.

[53] R. Bevans, *A quick guide to experimental design*, Dec. 2019. [Online]. Available: https://www.scribbr.com/methodology/experimental-design/.

[54] M. M. John, D. Gillblad, H. H. Olsson, and J. Bosch, "Advancing mlops from ad hoc to kaizen," in *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2023, pp. 94–101. DOI: 10.1109/SEAA60479.2023.00023.

[55] jcarter_tcgen, *Time to market (ttm) defined & why it's important*, Mar. 2024. [Online]. Available: https://tcgen.com/time-to-market/.

[56] J. Faustino, D. Adriano, R. Amaro, R. Pereira, and M. M. da Silva, "Devops benefits: A systematic literature review," *Software: Practice and Experience*, vol. 52, no. 9, pp. 1905–1926, 2022. DOI: https://doi.org/10.1002/spe.3096. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3096. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3096.

[57] *Payscale - salary comparison, salary survey, search wages.* [Online]. Available: https://www.payscale.com/en-eu/.

[58] S. Dasgupta and S. Hooshangi, "Code quality: Examining the efficacy of automated tools," 2017.

[59] T. Rukat, D. Lange, S. Schelter, and F. Biessmann, "Towards automated ml model monitoring: Measure, improve and quantify data quality," in *ML-Sys 2020 Workshop on MLOps Systems*, 2020. [Online]. Available: https://www.amazon.science/publications/towards-automated-data-quality-management-for-machine-learning.

[60] R. Ries, *What is mlops and how does it affect your models?* Oct. 2022. [Online]. Available: https://www.missioncloud.com/blog/what-is-mlops-and-how-does-it-affect-your-models#:~:text=How%20MLOps%20Affects%

20Performance%201%20Improved%20Data%20Usability,open%20workflow%20between%20operations%20and%20engineering%20teams.%20.

[61] M. Testi, M. Ballabio, E. Frontoni, G. Iannello, S. Moccia, P. Soda, and G. Vessio, "Mlops: A taxonomy and a methodology," *IEEE Access*, vol. 10, pp. 63 606–63 618, 2022. DOI: 10.1109/ACCESS.2022.3181730.

[62] A. Wiedemann, N. Forsgren, M. Wiesche, H. Gewald, and H. Krcmar, "The devops phenomenon," *Communications of the ACM*, vol. 62, p. 8, 2019. [Online]. Available: https://cacm.acm.org/magazines/2019/8/238341-research-for-practice-the-devops-phenomenon/fulltext.

[63] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, "Devops: Introducing infrastructure-as-code," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017, pp. 497–498. DOI: 10.1109/ICSE-C.2017.162.

[64] L. Zhu, L. Bass, and G. Champlin-Scharff, "Devops and its practices," *IEEE Software*, vol. 33, no. 03, pp. 32–34, May 2016, ISSN: 1937-4194. DOI: 10.1109/MS.2016.81.

[65] S. Fakhoury, Y. Ma, V. Arnaoudova, and O. Adesope, "The effect of poor source code lexicon and readability on developers' cognitive load," in *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, 2018, pp. 286–28 610.

[66] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research - an initial survey.," Jan. 2010, pp. 374–379.

[67] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*, English. Germany: Springer, 2012, ISBN: 978-3-642-29044-2. DOI: 10.1007/978-3-642-29044-2.

[68] P. W. M. Trochim, *Idea of construct validity*. [Online]. Available: https://conjointly.com/kb/construct-validity-idea/.

[69] M. A. García-Pérez, "Statistical conclusion validity: Some common threats and simple remedies," *Frontiers in Psychology*, vol. 3, 2012, ISSN: 1664-1078. DOI: 10.3389/fpsyg.2012.00325. [Online]. Available: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2012.00325.

[70] *Chatgpt*. [Online]. Available: https://chat.openai.com.