សាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ

ROYAL UNIVERSITY OF PHNOM PENH

ប្រធានបទនិក្ខេបបទ

Implementing MLOps Workflow and Define Role in an Industrial Setting

A Thesis
In Partial Fulfilment of the Requirement for the Degree of
Bachelor of Engineering in Information-Technology-Engineering

Hom Sokhim

June 2024

សាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ

ROYAL UNIVERSITY OF PHNOM PENH

ប្រធានបទនិក្ខេបបទ

Implementing MLOps Workflow and Define Role in an Industrial Setting

A Thesis
In Partial Fulfilment of the Requirement for the Degree of
Bachelor of Engineering in Information-Technology-Engineering

Hom Sokhim

Examination committee: Dr. Chhoeum Vantha
Mr. Chhim Bunchhun
Mr. Yok Tithrattanak
Mr. Chap Chanpiseth

June 2024

មូលន័យសង្ខេប

ការសិក្សាស្រាវជ្រាវមួយនេះ គឺយើងសិក្សាស្វែងរកការរួមបញ្ចូលគ្នានៃគម្រោងប្រតិបត្តិការរបស់ **Machine Learning Operation (MLOps)** និងការកំណត់នៅតួនាទីក្នុងការទទួលខុសត្រូវទៅលើការធ្វើគម្រោងក្នុង **Industrial** ឬ វិស័យឧស្សាហកម្ម។ ក្នុងនិក្ខេបបទនេះបញ្ហាញពីការពិនិត្យដ៏ទូលំទូលាយមួយនៃប្រធានបទ **MLOps, lifecycle, the DevOps lifecycle,** និងការបញ្ចូល **RACI (Responsible, Accountable, Consulted, Informed**) ទៅក្នុងដំណើរការ **MLOps** ។ ការសិក្សានេះពិនិត្យឡើងវិញនូវការពាក់ព័ន្ធនឹង **literature** ដើម្បីកំណត់តួនាទីនិងការទទួលខុសត្រូវរបស់អ្នកពាក់ព័ន្ធផ្សេងៗគ្នាដែលពាក់ព័ន្ធនឹង **MLOps Lifecycle** ផ្តល់ការយល់ដឹងអំពីទិដ្ឋភាពសំខាន់ៗនៃ **Machine Learning Operations.** ការវិភាគទៅលើរ **Architecture workflow, roles,** និង **MLOps architecture design** ពីប្រភពផ្សេងៗគ្នា ដោយផ្តល់នូវការយល់ដឹងដ៏មានតម្លៃចំពោះភាពស្មុគស្មាញនៃដំណើរការទាំងនេះ។ ក្នុងនិក្ខេបបទនេះគឺមានការស្រាយបញ្ហាញដ៏ទូលំទូលាយសម្រាប់ការយល់ដឹងអំពីភាពស្មុគស្មាញនៃ **MLOps** និងការរួមបញ្ចូលជាមួយការអនុវត្ត **DevOps** ។ វាគ្របដណ្តប់លើការពិនិត្យមើលការអនុវត្តគោលការណ៍ **DevOps** ទៅក្នុងដែននៃ **Machine Learning** ដោយបញ្ជាក់ពីអត្ថប្រយោជន៍និងបញ្ហាប្រឈមនៃការរួមបញ្ចូលនេះ។ ការអនុវត្ត **RACI** ការសិក្សាស្វែងយល់ពីក្របខណ្ឌ **RACI** និងការអនុវត្តរបស់វានៅក្នុង **MLOps lifecycle** ដោយកំណត់តួនាទី និងការទទួលខុសត្រូវរបស់អ្នកពាក់ព័ន្ធផ្សេងៗគ្នារួមទាំង **Business Stakeholders, Data Engineers, Data Scientists, Software Engineers, ML Engineers, and DevOps Engineers. End-to-End MLOps** បង្ហាញពីការរុករកលម្អិតនៃដំណើរការការងារ **MLOps** តាំងពីការចាប់ផ្តើមរហូរដល់ការបញ្ចប់ដែលគ្របដណ្តប់លើដំណាក់កាលនៃការបតិបត្តិការដូចជា **Business Understanding, Data Acquisition and Preparation, Model Development, Model Deployment, and Updating and Retraining. MLOps Architecture Design,** បង្ហាញពីសមាសធាតុសំខាន់ៗនៃ **MLOps Architecture** រួមទាំង **Data/feature store, model registry, model serving, and continuous integration and deployment (CI/CD) pipeline.** ការរកឃើញនៃនិក្ខេបបទនេះផ្តល់នូវការយល់ដឹងដ៏មានតម្លៃសម្រាប់ **Organizations** ដែលកំពុងស្វែងរកការអនុវត្ត **workflows** របស់ **MLOps** ប្រកបដោយប្រសិទ្ធភាព និងកំណត់តួនាទីនិងទំនួលខុសត្រូវរបស់ក្រុមរបស់ពួកគេនៅក្នុង **Industrial setting** មួយ។ ការគ្របដណ្តប់យ៉ាងទូលំទូលាយនៃប្រធានបទធ្វើឱ្យឯកសារនេះក្លាយជាធនធានដ៏មានតម្លៃសម្រាប់អ្នកអនុវត្ត និងអ្នកស្រាវជ្រាវក្នុងវិស័យ **Machine Learning Operations.**

Abstract

This thesis explores the integration of MLOps (Machine Learning Operations) workflows and the definition of roles within an industrial setting. The paper presents a comprehensive examination of the MLOps lifecycle, the DevOps lifecycle, and the incorporation of the RACI (Responsible, Accountable, Consulted, Informed) framework into the MLOps process. The study reviews relevant literature to define the roles and responsibilities of various stakeholders involved in the MLOps lifecycle, providing insights into the critical aspects of Machine Learning Operations. The analysis encompasses the architecture workflow, roles, and MLOps architecture design from different sources, offering valuable insights into the complexities of these processes. The document serves as a comprehensive resource for understanding the intricacies of MLOps and its integration with DevOps practices. It covers paper examines the application of DevOps principles to the domain of machine learning, highlighting the benefits and challenges of this integration. Applying RACI, The study delves into the RACI framework and its implementation within the MLOps lifecycle, defining the roles and responsibilities of different stakeholders, including business stakeholders, data engineers, data scientists, software engineers, ML engineers, and DevOps engineers. End-to-End MLOps, The paper presents a detailed exploration of the end-to-end MLOps workflow, covering the stages of business understanding, data acquisition and preparation, model development, model deployment, and updating and retraining. MLOps Architecture Design, The document outlines the key components of an MLOps architecture, including the data/feature store, model registry, model serving, and continuous integration and deployment (CI/CD) pipelines. The findings of this thesis provide valuable insights for organizations looking to effectively implement MLOps workflows and define the roles and responsibilities of their teams in an industrial setting. The comprehensive coverage of the topic makes this document a valuable resource for practitioners and researchers in the field of Machine Learning Operations.

Keyworks: Machine Learning, DevOps, MLOps, RACI Implementation in an industrial, Local Development, Server Development

SUPERVISOR'S THESIS SUPERVISION STATEMENT

TO WHOM IT MAY CONCERN

Name of program: Bachelor Program of Information Technology Engineering Name of candidate: HOM SOKHIM

Title of thesis report: Implementing MLOps Workflow and Define Role in an Industrial Setting

This is to certify that the candidate mentioned earlier completed the thesis for the above-titled bachelor's thesis report under my direct supervision. This thesis material has not been used for any other degree. I played the following part in the preparation of this thesis report: Implementing MLOps Workflow and Define Role in an Industrial Setting

Supervisor's name: Chap Chanpiset

Supervisor's signature:.....

Date: .....

CANDIDATE'S STATEMENT

TO WHOM IT MAY CONCERN

This is to certify that the thesis report that I (Hom Sokhim), hereby present entitled: "Implementing MLOps Workflow and Define Role in an Industrial Setting"

for the degree of Bachelor of Information Technology Engineering at the Royal University of Phnom Penh is entirely my own work and, furthermore, that it has not been used to fulfill the requirements of any other qualification in whole or in part, at this or any other University or equivalent institution.

This is to certify that the candidate mentioned earlier completed the thesis for the above-titled bachelor's thesis report under my direct supervision. This thesis material has not been used for any other degree. I played the following part in the preparation of this thesis report: Implementing MLOps Workflow and Define Role in an Industrial Setting

No reference to, or quotation from, this document may be made without the written approval of the author.

Signed by (Hom Sokhim):.....

Date: .....

Sign by supervisor: Chap Chanpiset

Supervisor's signature:.....

Date: .....

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my supervisor, Mr. Chanpiseth Chap, for his invaluable guidance, support, and expertise throughout the entire process of completing this Bachelor's degree thesis. His dedication, encouragement, and insightful feedback have been instrumental in shaping this research work.

I would also like to extend my appreciation to the faculty members of Royal University of Phnom Penh, for their continuous support and encouragement during my academic journey. Their wealth of knowledge and passion for teaching have greatly contributed to my growth as a researcher.

Furthermore, I am grateful to my family and friends for their unwavering love, understanding, and encouragement. Their constant support and motivation have been the driving force behind my success.

Lastly, I would like to express my sincere appreciation to all the participants who willingly contributed their time and expertise to this study. Without their participation, this research would not have been possible.

I am truly grateful to everyone who has played a part, no matter how big or small, in making this thesis a reality. Thank you all.

Contents

List of Figures

Chapter 1

1 Introduction

In the 21 century technology has developed to digital 4.0. Company enterprises and sectors move products to technology products to provide the customer with easy use of the products such as building Software, and Hardware[20] Nowadays some sectors are moving to AI(Artificial Intelligence)[8].
In the 21st century, technology has rapidly advanced, ushering in the era of digital 4.0. Companies, enterprises, and various sectors have embraced this digital transformation by shifting their focus from traditional products to technology-driven solutions[23]. This shift aims to provide customers with products that are easier to use, more efficient, and aligned with the digital age.
One significant aspect of this transformation is the development of software and hardware products. Companies are investing in the creation of intuitive software applications and user-friendly hardware devices that enhance the overall user experience. These products are designed to seamlessly integrate into people's lives, making tasks more convenient, efficient, and accessible.

However, the technological evolution does not stop at software and hardware. Nowadays, there is a growing trend of sectors exploring the potential of Artificial Intelligence (AI). AI, a branch of computer science, focuses on developing intelligent machines capable of performing tasks that typically require human intelligence. By leveraging AI technologies, sectors such as healthcare, finance, manufacturing, and transportation are revolutionizing their operations.

Artificial Intelligence (AI) The field of AI presents challenges in terms of precise definition. Alan Turing, in his influential paper 'Computing Machinery and Intelligence' [22] introduced the Turing test as a means to determine machine intelligence. According to this test, a machine can be considered intelligent if it can engage in conversation indistinguishably from a human, as judged by an impartial observer[11]. AI-powered systems can analyze vast amounts of data, identify patterns, and make predictions or decisions based on the available information. This has led to significant advancements in various areas. For instance, in healthcare, AI algorithms can assist in diagnosing diseases, analyzing medical images, and recommending personalized treatment plans. In finance, AI-powered chatbots can provide customer support and assist with financial planning. In manufacturing, AI-enabled robots can automate repetitive tasks, improving efficiency and productivity.

Instead, the term 'AI' is often used interchangeably with 'machine learning'. Machine learning encompasses algorithms and statistical models that learn from labeled training data [11]. By leveraging this training data, machine learning algorithms can recognize and infer patterns, enabling them to make predictions or decisions.

Machine learning (ML) is a subfield of computer science that has the broad objective of empowering computers to acquire knowledge and "learn" from data without explicit programming. It has an emphasis on practical objectives and real-world applications, with a particular focus on tasks such as prediction and optimization [3]. For example, Credit Card Fraud Detection using Decision Tree [18]. The decision tree algorithm [19] is a popular machine learning approach for fraud detection. It constructs a tree-like model of decisions based on historical data on fraudulent and non-fraudulent transactions. Relevant features are selected to distinguish between fraudulent and legitimate transactions. The algorithm recursively partitions the data based on these features, using criteria like Gini impurity or information gain [21]. Imbalanced data issues are addressed through techniques like oversampling or weighted loss functions. The trained decision tree can then predict the class label (fraudulent or legitimate) for new transactions. Model optimization techniques such as pruning or ensemble methods can improve performance. Typically, a Machine learning model that runs with the system in the real world needs to run in production, Machine learning in production will make an organization or company easy to manage, flexible input data, and retraining the model when changing the data and optimization model in real production [9]. Moreover, end-to-end processing in Machine learning allows the ML engineer can focus on all steps of Machine Learning processing that involve continued training and continued monitoring. When ML engineers need to manage, track, and make version updates to modify the model to better the real world in production[15]. For Machine Learning Model run in production is performed by MLOps as well.

MLOps is Abbreviations from Machine Learning Operation [17]. MLOps (Machine Learning Operation) is a concept that has an approach point or direct to the Machine Learning model. The Machine Learning model in production needs MLOps to manage and work on events of the ML model in production and make sure the End-to-End of the process stays on and stable. For a Model in a Machine Learning Operation to be successful, it must be defined as someone or teams who works on that model from End-to-End. In MLOps, each stage typically involves a person or a group responsible for its execution and management. For example, Model Deployment is the responsibility of the DevOps Engineer. The DevOps (Development Operation) Engineer is Responsible for deploying the trained model into production environments. They set up the necessary infrastructure, develop deployment pipelines, and ensure the model is deployed securely and efficiently. On the other hand, if there is no role to be assigned to each stage of model development, we will have a hard time determining who is responsible for each stage of the model creation or responsible for all stages of creation. How can we define roles clearly and correctly? MLOps has a lot of functions for responsibility and some functions need multiple people or teams to be responsible for it. So to define clarity and specificity we can do it on the RACI framework. RACI framework is used in project management to define roles and responsibilities clearly and easily to understand each role.

RACI, which stands for Responsible, Accountable, Consulted, and Informed, is a widely recognized tool in project management and organizational governance. Its primary purpose is to enhance clarity and establish clear lines of communication by defining roles and responsibilities within a team or organization. When applied effectively, the RACI matrix becomes an invaluable resource for project managers and team members alike. It helps prevent confusion and ambiguity regarding who is accountable for specific tasks or decisions, who should be involved in providing input or advice, and who needs to be kept informed of progress or outcomes.

This thesis aims to take a machine learning project and develop a prototype that applies MLOps practices to the machine learning lifecycle. It helps us to understand the problems and challenges when applying ML in an industrial environment and to better manage the entire machine learning lifecycle, which includes model and data versioning, keeping track of experiment results so that the machine learning project has reproducibility and traceability, model monitoring and model deployment in production. Furthermore, the thesis is implementing a RACI chart to clarify the responsibilities for each task. The RACI chart is a tool used to define and document roles and responsibilities in a project or organization. It stands for Responsible, Accountable, Consulted, and Informed. This will help to ensure clear ownership and accountability as the MLOps practices are implemented.

The rest of this thesis is organized as follows. In Chapter 2, the author conducts a literature review from other research papers and discusses the parts that are relevant to this thesis. Chapter 3 presents the background research that provides information about the DevOps Lifecycle, the principles of RACI, the End-to-End Machine Learning process, and the End-to-End MLOps approach. This chapter also includes the problem statement regarding the corresponding thesis topic, followed by the research questions. In Chapter 4, the author details the overall implementation of the topic and provides a cost estimate comparison between AWS (Amazon Web Service) and GCP (Google Cloud Platform). Chapter 5 discusses the topic of data storage, as well as the development of the machine learning model in both a local and server environment. The thesis concludes with Chapter 6, which provides an overall summary and conclusion. Finally, code examples are listed in the Appendix.

Chapter 2

2 Literature Review

Ravi Teja Yarlagadda. [24] The paper provides an in-depth exploration of the DevOps life cycle, highlighting the six key phases that form the foundation of DevOps practices. The phases outlined in the paper include planning, building, integration and deployment, monitoring, operating, and responding through feedback passage. Each phase is described in detail, emphasizing the activities and tasks involved in ensuring the successful development and deployment of software products.

Ioannis Karamitsos, Saeed Albarhami, and Charalampos Apostolopoulos. [13] discussed applying the DevOps practice of Continuous Automation for Machine Learning that discusses various challenges and approaches related to the integration of DevOps practices in the context of machine learning (ML) systems. It involves critical techniques such as DevOps Principles and Practices, Continuous Integration and Continuous Delivery (CICD), and Technical Debt in ML Systems.

Satvik Garg et al. [9] Machine Learning Operations (MLOps) has emerged as a crucial field of research in recent years, focusing on the deployment of AI models in production systems. The paper by Garg et al. (2022) provides insights into the different levels of MLOps maturity, namely MLOps Level 0, MLOps Level 1, and MLOps Level 2. At MLOps Level 0, manual workflows are prevalent, leading to sparse release iterations and a lack of automation. Moving to MLOps Level 1, automation is introduced to facilitate continuous model training and deployment, enhancing model prediction services. Finally, MLOps Level 2 involves the automation of a robust CI/CD pipeline for rapid and reliable updates to pipelines in production, incorporating six key CI/CD processes. The paper also highlights the importance of monitoring AI model performance and the challenges associated with frequent data changes.

Nipuni Hewage and Dulani Meedeniya. [12] The paper "Machine Learning Operations: A Survey on MLOps Tool Support" provides a comprehensive literature review on the comparison of various MLOps tools. The study explores the importance of Machine Learning Operations (MLOps) and discusses the functionalities and limitations of available platforms. The comparison of MLOps tools from the paper is shown in figure 1

TOOL NAME	Data Versioning	Hyperparameter Tuning	Model and Experiment Versioning	Pipeline Versioning	CICD availability	Model Deployment	Performance Monitoring
AWS SageMaker	✓	✓	✓	✓	✓	✓	✓
MLFlow	✓	✓	✓	✓	✓	✓	
Kubeflow		✓	✓		✓	✓	✓
DataRobot		✓	✓			✓	✓
Iterative Enterprise	✓		✓		✓	✓	✓
ClearML	✓		✓	✓	✓	✓	✓
MLReef	✓	✓	✓	✓	✓	✓	✓
Streamlit	✓		✓			✓	✓

Figure 1: MLOps tool comparison

Yizhen Zhao. [25] Provided about MLOps scaling and setting in the industry. This paper illustrates.

Machine Learning in Production: The document discusses the challenges and importance of implementing Machine Learning (ML) in a production environment. It emphasizes the need for ML systems to focus on engineering aspects beyond just implementing ML models.

Challenges in Machine Learning Projects: The challenges in ML projects are highlighted, including

the dependency level where different versions of data, code, and parameters can lead to different model behaviors. Managing these versions is crucial for maintaining control over the ML system.

MLOps Framework for Building Integrated ML System in Production: The document introduces the concept of MLOps, which is akin to DevOps for machine learning. It emphasizes the need for better collaboration and communication between data scientists and data engineers to automate the ML lifecycle and deploy models in production.

Data Version Control: The importance of data version control is discussed, with a focus on tools like DVC that enable data scientists to capture modifications, identify versions, and share datasets. The document highlights the need for handling large datasets and versions efficiently.

ML Model Deployment in Production Environment: The challenges and methods for deploying ML models in a production environment are explored. The document discusses the use of services like AWS Sagemaker Batch Transform for efficient model deployment.

Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. [14] The literature review from the provided document regarding Architecture and Workflow, as well as Roles and their intersections contributing to the MLOps paradigm.

Architecture and Workflow:

End-to-End MLOps Architecture: The document presents an end-to-end MLOps architecture that guides ML researchers and practitioners in setting up successful ML products. The architecture encompasses various stages, including MLOps product initiation, feature engineering pipeline, experimentation, and automated ML workflow pipeline up to model serving.

Technology-Agnostic Approach: The architecture is designed to be technology-agnostic, allowing flexibility for practitioners to choose the best-fitting technologies and frameworks for their specific needs. It emphasizes the use of both open-source tools and enterprise solutions, as well as the possibility of combining them for MLOps implementation.

Workflow Sequencing: The document outlines the sequence of tasks in different stages of the MLOps process, highlighting the importance of proper workflow orchestration, automation, and monitoring. It emphasizes the need for efficient management of tasks, artifacts, and metadata throughout the ML lifecycle

Roles and Intersections in MLOps:

Interdisciplinary Team Collaboration: The document emphasizes the importance of an interdisciplinary team approach in MLOps, where different roles intersect to design, manage, automate, and operate ML systems in production. Roles such as Data Scientists, Data Engineers, Software Engineers, DevOps Engineers, and Backend Engineers collaborate to ensure the success of MLOps initiatives. Figure 2

ML Engineer/MLOps Engineer Role:A key role identified is the ML Engineer or MLOps Engineer, who possesses cross-domain knowledge and combines skills from various disciplines. This role is responsible for building and operating the ML infrastructure, managing automated ML workflows, deploying models to production, and monitoring both the models and infrastructure.

Product-Focused Mindset: The document highlights the shift towards a product-oriented discipline in MLOps, where roles associated with designing ML products should have a product-focused perspective. This mindset is essential for successful MLOps implementation and aligning ML initiatives with business goals.

Collaborative and Cooperative Setup: The document stresses the need for teams to work collaboratively rather than in silos, fostering communication, knowledge sharing, and cooperation among team members. It emphasizes the importance of a group process in MLOps to ensure effective design, automation, and operation of ML systems.

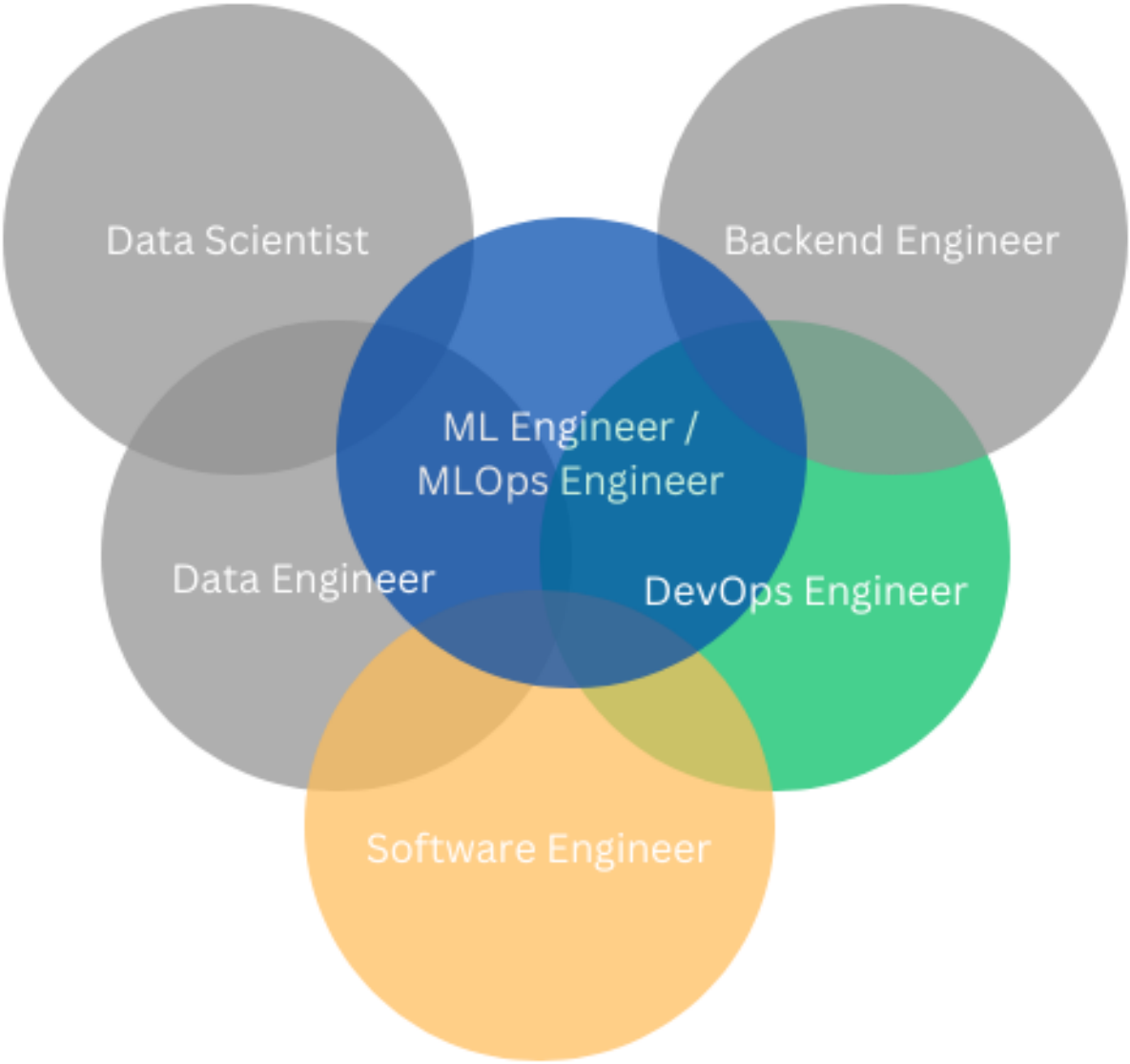

Figure 2: Roles and their intersections contributing to the MLOps paradigm.

Ioannis Karamitsos, Saeed Albarhami, and Charalampos Apostolopoulos. [13] The paper reviews the core principles of DevOps, including automation, continuous delivery, and fast feedback. The paper discusses various DevOps practices, such as continuous integration (CI), continuous delivery (CD), frequent deployments, QA automation, and collaboration.

Chapter 3

3 Background

3.1 DevOps Lifecycle

DevOps is a set of integrated activities or practices employed in automation and interlinking software development processes with IT developers with the aim of building, testing and releasing deliverables quickly and reliably.[24] DevOps life cycle focuses on loop practices for production aimed at quality and easy-to-automate deployment, monitoring, and maintain on the production. The processing DevOps practices on planning, coding, building, release, deploy operation, and monitoring.[7]

Planing phase: The planning aims to think about identifying the requirements of the resources and involves the problem statement. Resources are important to identify the resources needed to build the product. How many resources make our product work and what resources are needed for a product? DevOps already has plans for future product modifications that are necessary for the product because the product is modified as needed and improved.

Coding phase: In this phase, the developer writes code accordance with the requirement of the production that ensures code was created following the operation of the project. Developers also prepare code to formally apply algorithms or patterns that match the project to ensure the project has quality and efficiency.

Build phase: The build phase is a practice on the creation of software applications. typically It has activities such as compiling source code, running automation tests, and packaging the application for deployment. Who is that practice in this phase to ensure the software is quality and correctly ready for deployment to production? In another way, measuring the code is quick to change, and it is necessary when the software needs to be updated in the future.

Release phase: The release can start when the software is already checked. It has completed all requirements and testing software measures it is ready to go to production.

Deployment phase: In the deployment phase, we talk about bringing the product to the end user for use. It is the responsibility of the operation team that do activities such as setting up the environment for production building packages such as building CI CD, and setting up pipelines for production. The deployment needs a server for deploying the product that can run always for the user to use the product every time because the product that is deployed it user for the end user.

Monitor phase: In the monitor phase, a requirement the team uses techniques to gather metrics, log, and gain insights into the health of performance in the system. The team monitor also uses a monitoring tool to provide valuable feedback for improvement and optimization of the DevOps process.

3.2 Applying DevOps for Machine learning

Applying DevOps principles to machine learning (ML) projects can greatly enhance the efficiency, collaboration, and overall success of the research process. DevOps, which stands for Development and Operations, is an approach that emphasizes close collaboration between software developers and IT operations teams to streamline the development, deployment, and management of software systems. When applied to ML, DevOps can help automate and streamline the end-to-end ML research pipeline, enabling researchers to iterate quickly, deploy models more effectively, and facilitate collaboration among team members.

Version Control: Version control is a fundamental aspect of DevOps for ML. By utilizing a version control system (such as Git), researchers can track and manage changes made to ML code, configuration files, and datasets. This enables easy collaboration between team members, facilitates code review and

reproducibility, and ensures that historical versions can be accessed and analyzed.

Continuous Integration and Continuous Deployment (CI/CD): CI/CD practices involve automating the build, testing, and deployment of ML models. By setting up automated pipelines, researchers can integrate changes made by different team members, run tests, and deploy ML models to production or testing environments efficiently. CI/CD pipelines can help catch issues early, ensure code quality, and accelerate the research process.

Containerization: Containerization technologies like Docker provide a standardized and portable environment for ML research. By packaging ML code, dependencies, and configurations into containers, researchers can ensure that their experiments run consistently across different environments. Containers simplify deployment, enable reproducibility, and facilitate collaboration between researchers and operations teams. Here is some key that DevOps practice applies to Machine learning.

Monitoring and Logging: Monitoring and logging ML experiments are crucial for tracking performance, identifying issues, and ensuring model reliability. DevOps practices emphasize the use of robust monitoring systems to collect metrics, logs, and other relevant data. ML-specific monitoring tools, such as TensorFlow Data Validation or ModelDB, can help track model performance, data quality, and model drift over time.

3.3 Principle of RACI

RACI is an acronym derived from the four key responsibilities most typically used: responsible, accountable, consulted, and informed [4] It is used for clarifying and defining roles and responsibilities in cross-functional or departmental projects and processes. [5] There are a number of alternatives to the RACI model.

Responsibility(R) stands for the person or people who are responsible for their task. They will be doing the task and responsible for the task when they stay on their task. They also decide to complete and achieve the task.

Accountable(A): accountable is represented, when the task is ultimate. accountable are individuals to make decisions, define task success or failure, and approve the final task. It is accountable for the overall outcome or result of the task. however, accountability should be assigned to each task by only one person.

Consulted(C): The consulted has a role to play in providing advice, consulting on tasks, and also experts who can give advice and knowledge to decide to do a task successfully. knowledge, advice, and job counseling, but they do not have the right to make decisions and take responsibility for the task.

Informed(I): The informed is the role of informing about information of progress or outcome of a task for updating results, maintaining tasks, and executing tasks. they do not directly Consult, the responsibility and decisions on the task.

Task 1: Position A has R/A (Responsible/Accountable), Position B has C (Consulted), Position C has C, and Position D has I (Informed).

Task 2: Position A has R, Position B has I, Position C has R, Position D has I.

Task 3: Position A has R, Position B has A/R, Position C has C, Position D has C.

Task 4: Position A has I, Position B has A, Position C has R, Position D has I.

Task	Position A	Position B	Position C	Position D
Task 1	R/A	C	C	I
Task 2	R	I	R	I
Task 3	R	A/R	C	C
Task 4	I	A	R	I

R/A = Responsible and Accountable | R = Responsible | A = Accountable | C = Consulted | I = Informed

Figure 3: RACI Charting Example

3.4 End to End of Machine learning

Machine learning has made significant advancements in recent years, enabling the development of complex systems that can learn from data and make accurate predictions or decisions. One prominent approach in machine learning is the concept of end-to-end learning. End-to-end learning aims to automate the entire learning pipeline, from raw input data to the final output, without relying on handcrafted features or intermediate representations. This paradigm has gained considerable attention due to its potential to simplify and streamline the development of machine learning systems.

Traditionally, machine learning pipelines involved multiple stages[16], including data preprocessing, feature extraction, and model training. Each stage required human expertise and manual design choices, which could be time-consuming and error-prone. End-to-end learning, on the other hand, seeks to eliminate these manual steps and learn directly from the raw data, reducing the need for domain-specific knowledge and feature engineering.

The rise of deep learning, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), has greatly propelled the popularity of end-to-end learning. These deep learning architectures have demonstrated remarkable performance in various domains, such as computer vision, natural language processing, and speech recognition. By leveraging their ability to automatically learn hierarchical representations, deep neural networks can effectively extract features from raw data, making them suitable for end-to-end learning tasks.

End-to-end learning has been successfully applied in various domains. In computer vision, for example, end-to-end approaches have been used for tasks such as object recognition, image captioning, and visual question answering. Instead of manually engineering features or using intermediate representations, these systems directly process the raw pixel data and learn to extract relevant information for the specific task at hand.[1]

3.5 End-to-End MLOps

MLOps (Machine Learning Operations) is an emerging field that focuses on the deployment, management, and monitoring of machine learning models in production environments. It aims to bridge the gap between data science and operations, enabling organizations to effectively develop, deploy, and maintain machine learning systems at scale. Within the realm of MLOps, the concept of end-to-end MLOps has gained significant attention as it seeks to streamline and automate the entire machine learning lifecycle.

The traditional machine learning lifecycle involves distinct stages, including data collection and preprocessing, model training and evaluation, and model deployment and monitoring. Each stage requires different tools, technologies, and expertise, which can lead to challenges in integrating and managing the overall workflow. End-to-end MLOps aims to address these challenges by providing a holistic framework that encompasses all stages of the machine learning lifecycle.

One of the key objectives of end-to-end MLOps is to establish a seamless and efficient pipeline for deploying and managing machine learning models. This involves automating tasks such as data ingestion, feature engineering, model training, hyperparameter tuning, and model deployment. By automating these processes, end-to-end MLOps can significantly reduce the time and effort required to move models from development to production.

Another important aspect of end-to-end MLOps is ensuring the reliability and scalability of machine learning systems. This includes implementing robust monitoring and logging mechanisms to track model performance, detect anomalies, and facilitate troubleshooting. Additionally, end-to-end MLOps emphasizes the importance of version control and reproducibility, enabling organizations to track and reproduce model versions, configurations, and associated data.

End-to-end MLOps also addresses the need for collaboration and governance in machine learning projects. It provides mechanisms for effective communication and collaboration between data scientists, software engineers, and operations teams, facilitating knowledge sharing and ensuring smooth coordination throughout the development and deployment process. Moreover, end-to-end MLOps incorporates security and compliance considerations, ensuring that models and data are handled in a secure and compliant manner.

3.6 Problem statement

Machine Learning Operation (MLOps) has a lot of tasks to practice. Each task involves stakeholders to be responsible. An industrial has a lot of departments. Each department always has a team lead to manage the team. The problem when we do End-to-End of MLOps. So which team leads on each task?

- Implementing an MLOps (Machine Learning Operations) workflow - This involves defining the different stages and activities in the end-to-end MLOps lifecycle, such as data acquisition, model development, model deployment, and model monitoring/retraining.
- Defining the roles and responsibilities (using the RACI matrix) for the different stakeholders involved in the MLOps lifecycle - This includes roles like business stakeholder, data engineer, data scientist, ML engineer, and DevOps engineer, and their involvement (Responsible, Accountable, Consulted, Informed) in the various MLOps activities.
- Focusing on the implementation and integration of MLOps in an industrial/enterprise setting, rather than just a theoretical or academic context.

3.7 Research Question

- How does managing tasks play a crucial role in MLOps?

- What are the challenges typically encountered when deploying ML projects into production?

- Which services are commonly used in conjunction with MLOps?

- How does context relate to data, model, and application in the MLOps workflow?

- What distinguishes DevOps from MLOps?

Chapter 4

4 Methodology

The paper explores the integration of MLOps (Machine Learning Operations) workflows and the definition of roles within an industrial setting. The study reviews relevant literature to define the roles and responsibilities of various stakeholders involved in the MLOps lifecycle, providing insights into the critical aspects of Machine Learning Operations. The analysis encompasses the architecture workflow, roles, and MLOps architecture design from different sources, offering valuable insights into the complexities of these processes. The paper examines the application of DevOps principles to the domain of machine learning, highlighting the benefits and challenges of this integration. The study delves into the RACI framework and its implementation within the MLOps lifecycle, defining the roles and responsibilities of different stakeholders, including business stakeholders, data engineers, data scientists, software engineers, ML engineers, and DevOps engineers. The paper presents a detailed exploration of the end-to-end MLOps workflow, covering the stages of business understanding, data acquisition and preparation, model development, model deployment, and updating and retraining. The document outlines the key components of an MLOps architecture, including the data/feature store, model registry, model serving, and continuous integration and deployment (CI/CD) pipelines. The findings of this thesis provide valuable insights for organizations looking to effectively implement MLOps workflows and define the roles and responsibilities of their teams in an industrial setting.

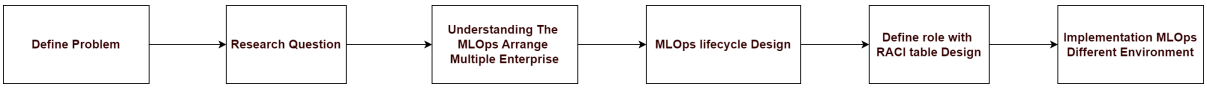

Figure 4: Methodology

4.1 Define Problem

4.2 Research Question

4.3 Understanding The MLOps Arrange Multiple Enterprise

4.4 MLOps lifecycle Design

The life cycle of MLOps explodes the process of machine learning model (ML) from the beginning to the production and continuing parts. The first phase aims to be a step-by-step process to ensure the functionality and modeling of the ML model. The practice of MLOps is shown in figure 5. To have a machine learning model in production, it is necessary to understand the purpose of putting it into production. In deploying the model into production, there are different designs according to the product and requirements of the company and according to the type of machine learning model. For running machine learning in production, there are usually some phases that are the same, such as phase development, deployment, and monitoring. These phases all come from DevOps. That DevOps is the base practice that marching learning follows. For DevOps, there is nothing different from MLOps. This section illustrates and presents the phase each step of the MLOps lifecycle.

4.4.1 Business Understanding and Requirement

Short MLOps for Machine Learning Operations run on tasks and tools used to facilitate the deployment of teams and machine learning models in production. It is a guideline for software engineering, engineering, and machine research to get a learning model of this machine that can be realized, scaled, and maintained. It should be a clear understanding and its requirements for a business.

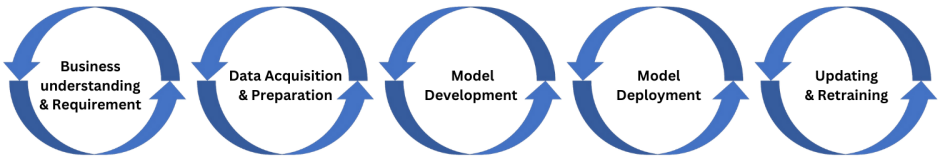

Figure 5: MLOps Lifecycle

Business understanding: Identify specific business goals and objectives that the machine learning model is expected to achieve. This can include improving the customer experience, streamlining operations, or increasing revenue. In the work performed by business people, they need to work with data science to create a machine learning model to match the business target. For example, the e-commerce company wants to input a recommendation system for products to increase sales and customers easily find the product in the application. So data science will create a recommendation system model to apply in production.

Requirement: Identify data requirements for training and deploying machine learning model. However, Identifying requirements has additionally some requirements like determining tools requirement for deploying machine learning models to production and implementing security measures to protect the deployed models, data, and infrastructure. This includes access controls, encryption, vulnerability management, and regular security audits. It's also required to adopt continuous integration and deployment (CI/CD) practices to automate the testing, validation, and deployment of machine learning models.

4.4.2 Data Acquisition and Preparation

Data acquisition and Preparation are an important phase in MLOps to collect data from any data source such as an internal database, third-party source, or synthetic data. The collection process may involve automated methods such as web scraping or using APIs to retrieve data programmatically. It can also involve manual data entry or data recording from physical sources. Ensure that the data is representative, relevant, and trained on high-quality and ensure data is ready for Model development and deployment. the data must be has quality and haven't noise that it effect to training data to be low-quality. This phase involves Data quality, Data preprocessing, and Data spitting. It's also can add Data privacy and security.

Data quality: Refer to identifying noisy data, handling missing values, outliers that need to check any missing value, data duplicated, data inconsistencies, or outliers that may affect the model's performance. Data cleansing techniques like outlier detection and data validation can be applied to improve data quality. Data quality will clean and remove noise in the data set for the model has a good performance.

Data preprocessing: refers to the steps taken to transform and prepare the data before it is used for training machine learning models in a production environment. Data preprocessing is a critical component of the MLOps workflow as it helps to ensure that the data is in a suitable format and quality for effective model training and inference. Data preprocessing involves Data Transformation, Feature

Encoding, Feature Scaling, Handling Imbalanced Data, Dimensionality Reduction, and Handling Time-Series Data. The data transformation includes converting variables to a suitable scale or distribution for modeling. Common techniques include normalization (scaling numerical features to a specific range), standardization (transforming features to have zero mean and unit variance), and log or power transformations to handle skewed data. The feature encoding categorical variables need to be encoded into numerical representations for machine learning models. Common encoding techniques include one-hot encoding, label encoding, and ordinal encoding. One-hot encoding creates binary variables for each category, label encoding assigns a unique numerical label to each category, and ordinal encoding assigns numerical labels based on the order or rank of the categories. In handling imbalanced data, the dataset may have imbalanced classes, where one class has significantly more data samples than others. To address this, techniques such as oversampling (creating synthetic samples of the minority class) or undersampling (reducing the number of samples from the majority class) can be applied to balance the distribution of classes.

Data Splitting: Divide the dataset into training, validation, and testing sets. The training set is the largest subset of the data and is used to train the machine learning model. It is used to learn the patterns and relationships in the data and adjust the model's parameters accordingly. The model is trained on this set in an iterative process to optimize its performance. The validation set is a smaller portion of the data that is used to fine-tune the model's hyperparameters and assess its generalization ability. Hyperparameters are configuration settings that are not learned from the data but affect the model's performance. By evaluating the model's performance on the validation set, you can make adjustments to the hyperparameters to improve the model's performance. The testing set is a separate portion of the data that is used to evaluate the final model's performance. It serves as an unbiased estimate of how the model will perform on unseen data. The testing set is used after model development and hyperparameter tuning to assess its ability to generalize to new, unseen data. It provides an unbiased evaluation of the model's performance and helps gauge its effectiveness in real-world scenarios.

4.4.3 Model Development

Model development focuses on the development process of the machine learning model. It refers to the process of creating, training, and refining marching learning models that will be deployed in a production environment. This phase involves several practices that will completed to ensure the success of the development machine learning model to be ready for deployment into production. First of all, data science or model developers choose an algorithm or model architecture that is suitable for the problem at hand. The consideration to be chosen for the machine learning model depends on factors such as the type of data, the size of the dataset, the nature of the problem, and the requirement. The algorithms such as decision tree, random forest, logistic regression, naive bayes, and K nearest neighbor. There are popular algorithms that are used for classification problems. Classification is supervised learning that is used for input data to prediction class [6]. More than this, there are other algorithms for unsupervised learning such as hierarchical clustering, k-means, mixture models, DBSCAN, and OPTICS [10]. They are all algorithms that will be considered for use in the machine learning model. Next, the stage of training the machine learning model, after the selected algorithm is already, the training stage uses the machine learning model to train the dataset. The training dataset involves adjusting its internal parameters to minimize the prediction error and feeding the input features target variables to the model. Feature variables also referred to as independent variables, predictors, or input variables, play a crucial role in predicting or explaining the target variable in a given context. These variables serve as the inputs or factors used to make predictions or draw conclusions about the output or dependent variable. After training the model, it goes to fine-tuning, hyper-parameter tuning is an approach to fine-tuning the model's hyper-parameter to optimize performance and configuration settings that are not learned from the data. After that, model evaluation, Is a process that is used to train model performance on the validation dataset. The process involves calculating and analyzing the relevant evaluated metrics such as accuracy, precision, recall, and F1-score. After the evaluation model. It practices on refinement model that based on the evaluation results, refines the model by iterating on the previous steps. This may involve adjusting hyperparameters, trying different architectures, applying regularization techniques, or enhancing feature engineering. The goal is to iteratively improve the model's performance until it meets the desired objectives. Finally, Model Validation, After finalizing the model, evaluate its performance on the testing dataset, which represents unseen data. This step provides an unbiased estimate of the model's performance and its ability to generalize to real-world scenarios.

4.4.4 Model Deployment

Model deployment refers to the process of making a trained machine learning model available for use in a production environment. It involves taking a model that has been developed and trained on a local machine or in a development environment and making it accessible to other systems, applications, or users. Model deployment has some practice depending on ML engineer practice. here is some practice for the deployment of the model:

Deployment Environment set up: Model deployment requires setting up the necessary infrastructure to host and serve the model. This may involve deploying the model on a cloud platform, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP), or on-premises infrastructure. On-premise servers are physical servers that are installed and maintained on-site at a company's location, rather than being hosted in a remote data center or cloud environment. These servers are typically used to store and manage data, run applications, and provide network services within an organization's internal network. On-premise servers require the company to purchase and maintain the necessary hardware, software, and infrastructure to support their operation. It also involves containerization (Docker). Docker can package the model and its dependencies into a container for consistent deployment across different environments. Simplifies deployment and management.

Model Serialization: This involves saving the trained model in a format that can be loaded and used by the deployment environment. Common formats include PMML (Portable Model Markup Language), ONNX (Open Neural Network Exchange), H5 file, PKL file, or TensorFlow SavedModel. Serialization ensures the model's knowledge is preserved and can be used for making predictions on new data.

Versioning: Assigning a version number to the deployed model is crucial. This allows you to track changes, revert to previous versions if issues arise, and manage different versions in production. Version control is essential for maintaining a history of your model and facilitating rollbacks if necessary. One common tool used for versioning in MLOps is Git. Git can be used to version control machine learning models, datasets, code, and other artifacts. Teams can create branches for different features or experiments, merge changes, track the history of modifications, and revert to previous versions if needed. By using Git along with platforms like GitHub, GitLab, or Bitbucket, teams can ensure reproducibility, collaboration, and traceability in their machine learning projects. Or other tools DVC (Data Version Control), or MLflow.

API Design: Models are typically deployed as web services, exposing an application programming interface (API) that allows clients to send input data and receive predictions or responses. Designing a well-defined and intuitive API is crucial for the usability and integration of the model into other systems.

Automating Deployment and Deployment Pipelines: Deployment Pipelines, Streamline the deployment process by creating a pipeline that automates tasks like packaging the model, deploying it to the chosen environment, and configuring the serving infrastructure. This reduces manual effort and minimizes the risk of errors during deployment. Popular tools for building deployment pipelines include MLflow and Kubeflow. Automating Deployment with Continuous Integration and Deployment (CI/CD). CI/CD practices can be applied to model deployment, enabling automated testing, building, and deploying of models whenever changes are made to the code or model artifacts. CI/CD pipelines help streamline the deployment process and ensure consistency.

Monitoring and Maintaining the Deployed Model: Performance Monitoring, Track metrics like accuracy on a set of validation images to ensure the model continues to perform well. You might also monitor latency (prediction speed) to optimize user experience. Alerting, Set up alerts to notify you if the model's accuracy drops significantly. This could indicate data drift (changes in the distribution of images the model encounters) and might necessitate retraining.

Model deployment also considering on Security and Scalability. Security, Implement security measures to protect the model from unauthorized access and potential manipulation. This might involve authentication for API access. Scalability, If your application experiences a surge in usage, the deployment environment should be scalable to handle the increased load. Cloud platforms offer easy scaling options.

4.4.5 Updating and Retraining

In MLOps (Machine Learning Operations), updating and retraining models are crucial steps to ensure that your machine learning models stay accurate and up-to-date. In the world of MLOps, keeping your deployed machine learning models performing optimally requires constant attention. Updating and retraining are two crucial practices that ensure your models adapt to changes and deliver consistent value. An overview of updating and retraining in MLOps.

Monitoring and Evaluation: To determine when to update or retrain a model, it's important to monitor its performance and evaluate its effectiveness on an ongoing basis. This can involve tracking metrics such as accuracy, precision, recall, or other relevant performance indicators. Monitoring can be done by setting up automated processes that periodically evaluate the model's performance against a predefined set of criteria.

Data Collection and Preparation: Before updating or retraining a model, you need to collect new data or additional labeled data. This can involve various methods, such as gathering data from new sources, getting feedback from users, or utilizing active learning techniques. Once you have the new data, you need to preprocess and prepare it for training, ensuring it is clean, consistent, and compatible with the model's input requirements.

Updating the Model: If the changes in the data distribution are minimal or the model's performance is satisfactory, you may choose to update the model rather than retraining it from scratch. Updating typically involves using the new data to fine-tune the existing model, allowing it to adapt to the changes in the data. Fine-tuning techniques like transfer learning or online learning can be employed to update the model efficiently.

Retraining the Model: In cases where the changes in the data distribution are significant, or the model's performance has degraded below acceptable thresholds, retraining the model from scratch may be necessary. Retraining involves using the combined existing and new data to train a new model. This process can include selecting an appropriate algorithm, hyperparameter tuning, and training the model on a suitable compute infrastructure.

Versioning and Tracking: It's essential to maintain a version control system for your models, datasets, and associated code. This allows you to track changes, compare performance between different versions, and roll back to previous versions if necessary. Versioning can be done using tools like Git, and you can also utilize metadata tracking systems to record information about the training data, hyperparameters, and evaluation metrics for each model version.

Automation and Deployment: To ensure a streamlined and efficient updating and retraining process, it's beneficial to automate these steps as much as possible. This can involve setting up pipelines or workflows that automatically trigger updates or retraining based on predefined criteria, integrating monitoring and evaluation into the pipeline, and automating the deployment of updated models to production environments.

Testing and Validation: After updating or retraining a model, thorough testing and validation are necessary to ensure that the updated model performs as expected. This can involve using a separate validation dataset or conducting A/B testing to compare the performance of the updated model against the previous version or other baselines. Rigorous testing helps identify any potential issues or regressions before deploying the updated model to production.

4.5 Define role with RACI table Design

The process of Machine Learning Operations (MLOps) encompasses several stages, each involving different individuals and teams. Defining clear roles and responsibilities within these stages can be challenging due to the complexity and interdisciplinary nature of MLOps. To provide a clearer understanding of the division of labor, one effective approach is to utilize the RACI framework and create a RACI diagram.

The RACI framework helps define the roles and responsibilities of individuals or groups involved in a process. RACI stands for Responsible, Accountable, Consulted, and Informed. Each role is assigned one of these four designations, clarifying their level of involvement in a specific task or decision. Figure 6 shows RACI charting that task design that is involved in MLOps.

Task	Business stakeholder	Subject matter expert	Data engineer	Data scientis	Software engineer	ML engineer	DevOps engineer
1. Problem definition & requirement	R/A		C	C		C	
2. Exploratory data analysis		R		R		C	
3. Implementation		R	A/R	C	C	C	
4. Feature engineering		C	A	R		C	
5. Experiment		C		R/A		R	
6. Model training and evaluation	I	R	I	R/A		C	
7. Model Package and Deployment	I	I	I	C	C	R/A	R
8. Monitoring and Maintenance	I	I		I	I	R/A	R
9. Retraining and Iteration		C	C	R	I	R/A	

R/A = Responsible and Accountable | R = Responsible | A = Accountable | C = Consulted | I = Informed

Figure 6: RACI TABLE

4.5.1 Business Stakeholder

Business stakeholders are **responsible** and **accountable** for the first task, It refers to the problem definition and requirement. In this role, Business Stakeholders work with Data engineers and Data Science to complete this task and make decisions for continuing to the next task.

Problem definition: Understanding business problems. Start by clearly understanding the business problem or opportunity that the ML project aims to address. Work closely with domain experts to define the problem statement, its context, and the desired outcomes. Next, Identify the key performance indicators (KPIs) that will measure the success of the ML model in addressing the business problem. These metrics could include accuracy, precision, recall, customer satisfaction, revenue impact, or any other relevant metrics.

Requirement: Business stakeholders work with data science and data engineers to understand the data requirements for the ML model. Specify the types of data needed, their sources, formats, and any data quality or privacy considerations, and collaborate with data scientists to define the expected behavior of the ML model. Discuss any constraints, such as latency requirements, resource limitations, or regulatory compliance that need to be considered during model development and deployment.

Business stakeholders are **informed** on tasks 5, 6, and 7. When these tasks are running the data science and ML engineer will inform Business stakeholders to get information about Model training, Evaluation, Model deployment, Monitoring, and Maintenance.

Business stakeholders typically won't need in-depth technical information about every step of Model training, Evaluation, Model Deployment, Monitoring, and Maintenance (MLOps lifecycle). However, keeping them informed at key points is crucial for successful project management and alignment. Here's a breakdown of how data science and ML engineers can effectively communicate with business stakeholders for each MLOps stage.

Model Training: Inform stakeholders about the training progress, including estimated timelines and any significant challenges encountered (e.g., data quality issues). Focus on Business Impact, Highlight how the training process is progressing towards achieving the defined business goals.

Model Evaluation: Performance Metrics, Share key performance metrics (KPIs) after evaluation, explaining how well the model performs against the success criteria established during problem definition. Actionable Insights, Provide clear insights derived from the evaluation, such as areas for improvement or potential use cases for the model.

Model Deployment: Deployment Plan Overview, Briefly explain the deployment plan, outlining where the model will be deployed (production environment, testing environment) and its expected impact on business processes. Success Measurement, Discuss how the success of the deployed model will be measured, aligning with the KPIs defined earlier.

Monitoring and Maintaining: Regular Performance Reports, Business Impact Updates: Quantify the ongoing business impact of the deployed model, showcasing its value to the organization. Provide periodic reports on the model's performance in production, highlighting any deviations from expected behavior or emerging issues. Business Impact Updates, Quantify the ongoing business impact of the deployed model, showcasing its value to the organization.

4.5.2 Data Engineer

First of all the Data engineer **consulted** on the first task, Problem definition, and the requirement that Data engineering work with Business stakeholders and Data science consultants about the problem to get insight for collection data for the machine learning model. The Data engineer then collaborated with the data science team to determine the best approach for collecting and preparing the necessary data. Together, they identified key data sources, established data pipelines, and ensured data quality and integrity throughout the process. This close partnership between Data engineering, data science, and Business stakeholders was crucial in developing a successful machine learning model that could provide valuable insights and drive business decisions.

Data Engineers are indeed **accountable** and **responsible** for implementation in task 3, which involves the entire data pipeline of collecting, storing, processing, and maintaining data quality. This stage lays the groundwork for successful machine learning projects by ensuring the data used to train models is clean, accurate, and accessible.

Data Collection: Identify and implement methods to collect data from various sources. This may involve working with APIs, databases, or even manually uploading data files. Ensure data collection adheres to security and privacy regulations.

Data Storage: Design and implement data storage solutions that are scalable, reliable, and secure. Common data storage options for MLOps include data warehouses, data lakes, and cloud storage platforms. Consider factors like data volume, access needs, and cost when choosing a storage solution.

Data Processing: Develop and implement data pipelines to automate the process of moving data from its source to its destination (data storage). These pipelines may involve data transformation tasks to prepare the data for further analysis. Data processing ensures data consistency and reduces the risk of errors.

Data Quality Maintenance: Implement data quality checks to identify and address issues like missing values, inconsistencies, and outliers. Data quality monitoring tools can be helpful in this process. Regularly monitor data quality to ensure it meets the defined standards for machine learning purposes.

Data Engineer **accountable** for task 4 Feature engineering, In this task, the Data engineer will decide to allow features of data for data science to get the data to implementation with a machine

learning model. The Data Engineer plays a critical role in task 4, which is feature engineering. In this stage, the data engineer prepares the data for the data science team by creating features that will be most useful for training the machine learning model.

Data engineers are **informed** on Model training, evaluation, and deployment. While data engineers are informed about these stages, the depth of their involvement will vary.

Model Training: They might not directly participate in training the model, but their work on data pipelines ensures a steady flow of high-quality data for training.

Model Evaluation: They might be involved in analyzing evaluation results to identify data-related issues that could be impacting performance.

Model Deployment: They might play a more active role in deploying the model by configuring data infrastructure and ensuring the model has access to the required data in production.

4.5.3 Data scientist

Data scientists are indeed **consulted** at the very beginning of the MLOps process, specifically in task 1: Problem Definition and Requirement Gathering. Their expertise is crucial in setting the stage for a successful project.

Understanding the Business Problem: They work closely with business stakeholders to gain a deep understanding of the specific business challenge or opportunity the project aims to address. This might involve tasks like identifying pain points or areas for improvement, Understanding how machine learning could potentially solve the problem, and Defining success metrics to measure the impact of the model.

Defining Initial Requirements: They work with the project team to define the initial requirements for the machine learning model, such as Accuracy requirements, Explainability (ability to understand model decisions), Latency (speed of model response), Scalability (ability to handle increasing data volumes). Effective problem definition and requirement gathering require collaboration between the data scientist, data engineer, and business stakeholders.

Data scientists are absolute **responsible** for task 2: Exploratory Data Analysis (EDA) in the MLOps lifecycle. EDA is a critical stage where the data scientist dives deep into the data to understand its characteristics, identify patterns, and prepare it for model training. EDA involves practicals such as Data Acquisition, Data Cleaning and Preprocessing, Exploratory Analysis, Feature Engineering, and Data Quality Assessment [2]. Data Acquisition, Collaborate with data engineers to access the data prepared in task 3 (data pipeline and storage). Explore and understand the data structure, format, and content. Data Cleaning and Preprocessing, Identify and address missing values, inconsistencies, and outliers in the data. This might involve techniques like imputation, data deletion, or outlier wansorization and Performing data normalization or scaling to ensure features are on a similar scale, improving model performance. Exploratory Analysis, Use statistical methods and data visualization techniques to understand the distribution of features, identify potential relationships between variables, and uncover any hidden patterns in the data Common techniques include calculating summary statistics, and creating histograms, scatter plots, and boxplots. Feature Engineering, Based on the EDA findings, the data scientist might start initial feature engineering tasks. This could involve feature selection (choosing the most relevant features) or creating new features by combining existing ones.

Data scientists **consulted** on task implementation at task 3. There can be some consultation between data scientists and data engineers during task 3 (data pipeline implementation). While data engineers take the lead in building and maintaining the data pipeline, data scientists can provide valuable input such as Data Understanding, and Feature Engineering Considerations. Data Understanding, Data scientists have a deep understanding of the data required for the model they will eventually build. They can collaborate with data engineers to identify relevant data sources. It points data engineers toward the specific data sources needed for training the model and defines data quality requirements. It Specifies the acceptable levels of missing values, outliers, and inconsistencies for the data used in model training. Feature Engineering Considerations, During data pipeline design, the data scientist can share initial thoughts on feature engineering. This might involve such as identifying Potential Features, and Data Transformation Needs. Identifying Potential, features suggesting features in the raw data that

might be relevant for the model. Data Transformation Needs, Providing insights into potential data transformations that might be required during feature engineering (task 4).

Data scientists **consulted** on task model package and deployment at task 7. While data scientists aren't solely responsible for model packaging and deployment (task 7) in MLOps, they are definitely consulted and involved in the process. Data scientists collaborate with ML engineers to ensure the model is prepared for deployment.

Data scientists are indeed **responsible** for Feature Engineering at task 4. Feature Engineering in the MLOps lifecycle. Feature engineering is a crucial stage where the data scientist transforms the raw data into features that are most useful for training a machine learning model. They utilize the findings from Exploratory Data Analysis (EDA) in task 2 to guide their feature engineering decisions. This includes understanding the data distribution, relationships between variables, and potential issues that need to be addressed. After they get insight from task 2 (EDA), They do the Feature selection that is to choose the most relevant and informative feature from the available data. irrelevant or redundant features can negatively impact model performance. Data science also does the feature creation that creates new features by transforming or combining existing ones. This can involve tasks like Binning grouping continuous data into discrete categories, Encoding converting categorical data into numerical values and Deriving new features creating new features based on domain knowledge or mathematical calculations.

Data Scientists are indeed **responsible and accountable for task 5:** Experimentation in the MLOps lifecycle. This stage involves training, evaluating, and iteratively improving the machine-learning model. Data scientists take the lead in this crucial task such as Model Training Expertise, Hyperparameter Tuning, Model Evaluation, and Interactive improvement. Data scientists lead experimentation, and collaboration with Data engineers to ensure a steady flow of high-quality data for training and experimentation. ML Engineer might be involved in building and deploying the model training pipeline, especially in complex MLOps projects. And with business Stakeholders to provide feedback on the model's performance from a business perspective and help define success criteria.

Accountability for Results: Data scientists are accountable for the outcome of the experimentation stage. They need to ensure the developed model meets the defined success metrics and delivers value to the business.

Data scientists informed at task Monitoring and Maintenance task 7. 7.

4.5.4 Software Engineer

Software engineers are indeed consulted throughout the MLOps lifecycle, particularly in task 3: Implementation (data pipeline) and task 7: Model Packaging and Deployment. While they don't lead these tasks entirely, their expertise is crucial for successful implementation and deployment of machine learning models.

Task 3: Implementation (Data Pipeline): Software engineer collaborates with Data engineer to work closely with data engineers to design and implement the data pipeline. They might contribute by Choosing Appropriate Tools and Technologies, Automating Data Pipelines, and Implementing Security Measures.

Task 7: Model Packaging and Deployment: Software engineer collaborates with ML Engineers to play a supporting role alongside ML engineers during model packaging and deployment.

4.5.5 ML Engineer

There can be some consultation between ML engineers and other team members in the initial stages of MLOps (tasks 1-4). While they might not be the sole lead, their expertise in machine learning engineering can be valuable during these phases.

Task 1: Problem Definition and Requirement Gathering Technical Feasibility: ML engineers can provide insights into the technical feasibility of using machine learning for the identified business

problem. They can assess factors like data availability, model complexity, and computational resources needed.

Task 2: Exploratory Data Analysis (EDA) Tool Selection: ML engineers can suggest specific data analysis tools or libraries that might be helpful for EDA based on the type of data and the desired analysis.

Task 3: Implementation (Data Pipeline) Scalability Considerations, ML engineers can advise on building a data pipeline that is scalable and can handle the potential growth of data volume over time, and Infrastructure Planning, They might collaborate with data engineers on infrastructure planning for data storage and processing, considering the needs of model training and deployment.

Task 4: Feature Engineering Feature Engineering Techniques, ML engineers can share their knowledge of specific feature engineering techniques that might be suitable for the machine learning model being built, and Automation Considerations, They might offer insights into automating certain aspects of feature engineering to improve efficiency.

ML engineers are indeed **responsible** for task 5: Experimentation in the MLOps lifecycle. This stage involves training, evaluating, and iteratively improving the machine-learning model. ML engineers take the lead in this crucial task:

Machine Learning Expertise: ML engineers possess the skills and knowledge to select, implement, and optimize machine learning algorithms for the specific problem. They can choose algorithms that align with the data, desired outcomes, and computational resources available.

Model Training Pipelines: They design and build automated pipelines for training the model. These pipelines can handle tasks like data pre-processing, model training, hyperparameter tuning, and model evaluation.

Hyperparameter Tuning: ML engineers optimize the model's hyperparameters, which are settings that control the learning process. Tuning these parameters significantly impacts the model's performance.

Model Selection and Evaluation: They collaborate with data scientists to select appropriate evaluation metrics based on the problem and business goals. They then implement these metrics to assess the model's effectiveness.

Deployment Considerations: During experimentation, ML engineers consider deployment requirements and ensure the model can be packaged and integrated into the production environment effectively.

ML engineers are responsible and accountable for model package and deployment, monitoring and maintenance, and retraining and iteration in MLOps. While there might be some overlap or collaboration with other team members in these tasks, ML engineers take the lead and ensure these stages are completed successfully.

4.5.6 DevOps Engineer

DevOps engineers also play a crucial role in model package and deployment, as well as monitoring and maintenance in the MLOps process. It depends on the specific team structure and project requirements within an organization. Traditionally, there can be some overlap in responsibilities between MLOps engineers and DevOps engineers, particularly in tasks related to deployment and monitoring.

Model Package and Deployment: DevOps engineers are responsible for the packaging and deployment of machine learning models into production environments. They work closely with ML engineers to ensure that the models are packaged in a deployable format, such as Docker containers or serialized files. DevOps engineers set up the necessary infrastructure, configure deployment pipelines, and automate the deployment process. They ensure that the models can be easily deployed, scaled, and integrated with existing systems.

Monitoring and Maintenance: DevOps engineers are responsible for monitoring the performance and behavior of deployed machine learning models in production. They set up monitoring systems

and tools to collect and analyze relevant metrics, such as prediction accuracy, response time, resource utilization, and system health. DevOps engineers also handle the maintenance of the deployed models, including managing software updates, infrastructure scaling, and troubleshooting issues that may arise. They collaborate with ML engineers and operations teams to ensure the models are operating optimally and meet the required service-level agreements (SLAs).

4.6 Implementation MLOps Difference environment

Chapter 5

5 Project Implementation

5.1 MLOps Architecture Design

This architecture illustrates the end-to-end process of deploying a machine learning model into production. It starts with data storage, which feeds into a local model development environment. This local environment handles tasks like exploratory data analysis, data validation, data preparation, model training, evaluation, and validation, before exporting the finalized model.

The architecture then showcases the remote model development server, which automates the same pipeline of data preparation, model training, evaluation, and validation. This allows for iterative model updates and optimizations based on performance monitoring.

Once the model is ready, it is packaged and stored in a central repository or distribution system. From there, the model is integrated into an API that can be accessed by the end-user application, such as Postman or an Amazon API Gateway.

The overall workflow emphasizes the importance of a streamlined CI/CD (Continuous Integration/Continuous Deployment) process to efficiently push the machine learning model into production, leveraging automation tools and techniques. This ensures the model can be quickly updated, optimized, and made available to end-users in a reliable and scalable manner.

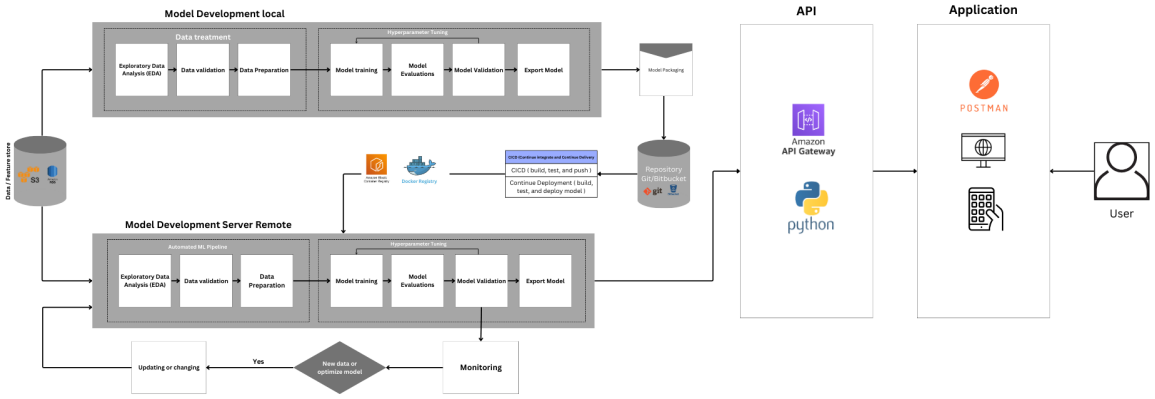

Figure 7: MLOps architecture

5.1.1 Data / Feature store

In this component, Data stores are used for various purposes beyond machine learning, such as data warehousing, business intelligence, and analytics. Popular data stores include relational databases (MySQL, PostgreSQL), data lakes (AWS S3, Azure Data Lake Storage, AWS RDS, etc.), and data warehouses (Snowflake, Redshift, AWS Glue, etc). It can hold various data formats, including structured data (tables in databases), unstructured data (text, images, videos), and semi-structured data (JSON, XML).

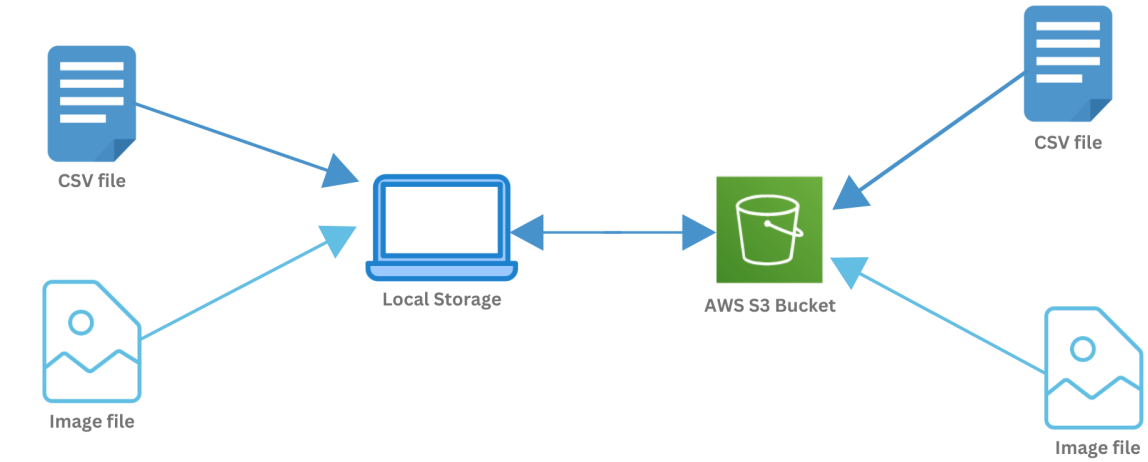

Figure 8: Overview Data Storage

A data store is a general-purpose repository for housing raw data used to train and potentially evaluate machine learning models.

Feature Store, Feature stores are particularly valuable for managing features throughout the machine learning lifecycle, including training, serving, and retraining models. Feature stores typically focus on structured, preprocessed data ready for model training. They might not be optimized for handling raw data or unstructured data. A feature store is a specialized repository designed specifically for storing, managing, and serving features used in machine learning models.

For this research project, we store data in different locations based on the size of the dataset. For the smaller datasets, we store the raw data, training data, and testing data locally on the development machine. Each of these smaller datasets has a size of approximately 111 KB (kilobytes).

However, for the larger datasets, we utilize cloud-based storage solutions. Specifically, we store a 9 GB (gigabyte) dataset of 250,000 JPG images on the Amazon S3 cloud storage service. This allows us to work with the larger dataset when developing and training our machine learning models, either on a remote server or within a Sagemaker notebook instance.

By separating the data storage based on size, we can optimize our workflow. The smaller datasets are easily accessible on the local development machine, while the larger dataset is securely stored in the cloud and can be accessed as needed when training more complex machine learning models.

5.1.2 Model Development Local

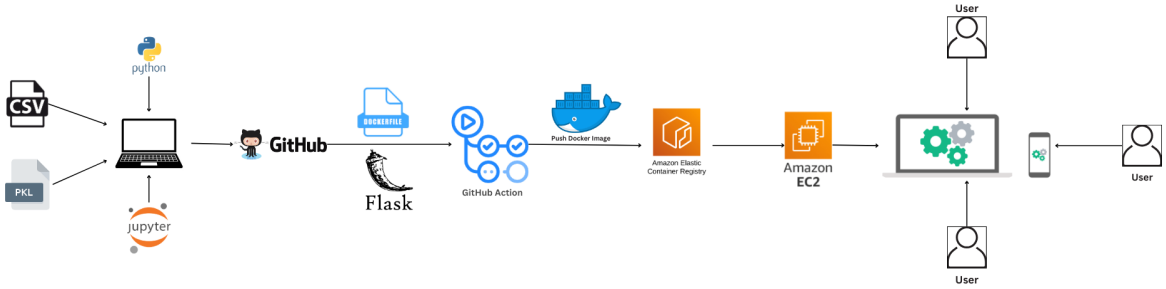

Figure 9: Local Development Workflow

This local environment handles tasks like exploratory data analysis, data validation, data preparation, model training, evaluation, and validation before exporting the finalized model. We used a local environment to manage this end-to-end workflow of the MlOPs pipeline. A local environment to manage this end-to-end workflow is crucial for ensuring the quality and reproducibility of the machine learning projects. It allows us to iterate quickly, experiment with different approaches, and ultimately deliver a high-performing model that can be seamlessly integrated into production systems.

Exploratory Data Analysis (EDA): This involves carefully examining the dataset to understand its characteristics, identify patterns, detect issues, and inform feature engineering and modeling decisions.

Data Validation: Ensuring the data is clean, complete, and representative of the real-world problem you're trying to solve. This includes handling missing values, outliers, data types, etc.

Data Preparation: Transforming the raw data into a format suitable for modeling, including feature engineering, scaling, encoding categorical variables, etc.

Model Training: Applying machine learning algorithms to the prepared data to train predictive models.

Model Evaluation and Validation: Assessing the performance of trained models using appropriate metrics, and validating their generalization to new, unseen data.

Model Export: Packaging the finalized model in a format that can be deployed and used for inference in production.

For the implementation, we utilized several machine learning algorithms such as Linear Regression, Lasso, Ridge, K-Neighbors Regressor, Decision Tree, Random Forest Regressor, XGBRegressor, CatBoostingRegressor, and AdaBoost Regressor. We trained all the algorithms on a dataset with a size of 111 kilobytes (KB). The hardware used for this work was a laptop with an Intel(R) Core(TM) i7-10750H CPU, running at 2.60GHz with 6 cores, 8 Gigabytes (GB) of RAM, and a 500GB Solid-State Drive (SSD).

The objective of this work was to develop a model for predicting student scores, with the aim of identifying reading and writing scores based on the input values of students' reading and writing scores. After building the model, we exported it to a file named "model.pkl" for use in web applications.

For the web application, we used the Python Flask framework to build an API and a web client. We then pushed all the code and packages to a GitHub repository, enabling the use of GitHub Actions for Continuous Integration and Continuous Deployment (CI/CD) to an AWS EC2 instance server.

By deploying the application to the AWS EC2 instance, we can now allow client users or end-users to access and use the web application. Additionally, we can monitor the server for tracking and maintenance purposes, ensuring the smooth operation of the system. The architecture workflow show in figure 9.

5.1.3 Model Development in Server Remote

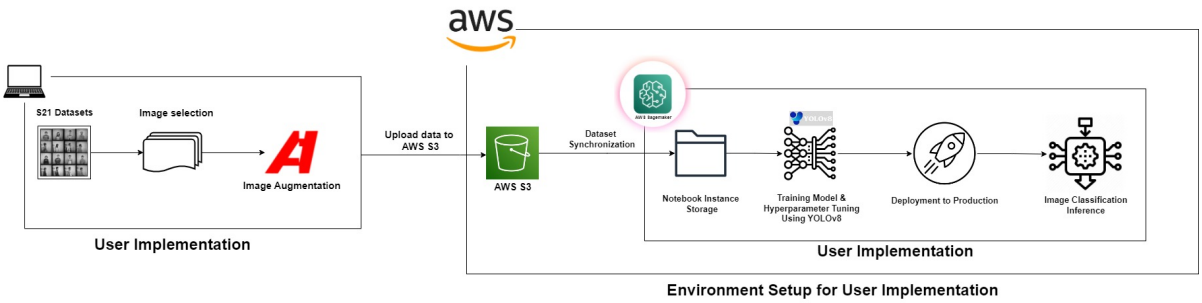

Figure 10: Server Development Workflow

The architecture diagram illustrates the end-to-end workflow for deploying a machine learning model on Amazon Web Services (AWS). At the start of the process, the user begins with a set of 21 datasets and selects relevant images from them. To expand the dataset, the user performs data augmentation, which involves applying various transformations to the images to increase their diversity and size.

With the augmented image data, the user then uploads it to the AWS S3 storage service. S3 serves as the central repository for storing the user's image data. The next step is dataset synchronization, where the image data stored in S3 is synchronized with the user's local computing environment, ensuring seamless access to the data.

The user then sets up a notebook instance on AWS, which provides a computing environment for further model development and training. Within this notebook instance, the user trains a machine learning model, specifically a YOLO (You Only Look Once) object detection model, and performs hyperparameter tuning to optimize the model's performance.

After the model training and tuning are complete, the user deploys the trained model to a production environment. This production deployment allows the model to be used for image classification inference tasks, where the model can make predictions on new, unseen images.

The architecture diagram clearly outlines the flow of data and the various AWS services utilized throughout this machine learning model deployment process. It provides a comprehensive overview of the steps involved, from initial data preparation to final model deployment, highlighting the key components and their interconnections within the AWS ecosystem.

5.2 Cost Estimating GCP and AWS

In this section, we will analyze and compare the monthly computing costs between two major cloud service providers, namely AWS (Amazon Web Services) and GCP (Google Cloud Platform). shown in figure 11. The comparison will be based on various instance types offered by both providers, ensuring that each instance type has equivalent specifications in terms of CPU, RAM, and Region. By examining the cost differences across these comparable instances, we can gain insights into the relative affordability and value offered by each platform. Before we delve into the comparison, let's take a moment to understand the key details about each cloud service provider and their respective server offerings.

5.2.1 GCP (Google Cloud Platform)

Google Cloud Platform (GCP) is a cloud computing platform provided by Google that offers a wide range of infrastructure and platform services for building, deploying, and scaling applications. Vertex AI is a specialized service within GCP that focuses on machine learning and artificial intelligence (AI) capabilities. It provides a unified platform for developing, deploying, and managing machine learning models. [1]

5.2.2 AWS (Amazon Web Service)

Amazon Web Services (AWS) is a comprehensive cloud computing platform provided by Amazon. It offers a wide range of cloud services that enable businesses and developers to build, deploy, and manage applications and IT infrastructure with scalability, reliability, and security. Amazon SageMaker is a fully managed machine learning service provided by Amazon Web Services (AWS). It enables developers and data scientists to build, train, and deploy machine learning models at scale. SageMaker offers a comprehensive set of tools and services that cover the entire machine-learning workflow. [2]

Comparison GCP and AWS

Service	CPU	RAM	Region	Estimated costs
GCP (Google cloud computing)	4 core	16g	Singapore	$163.20/month
AWS (Amazon web service)	4 core	16g	Singapore	$ 212.4/month

Figure 11: Compatison GCP and AWS

In figure 11, When choosing instance types for machine learning workloads, it's important to consider the cost-effectiveness of the services offered by different cloud providers. In this case, you've compared the pricing for a 4-core CPU and 16 GB RAM instance in the Singapore region between Google Cloud Platform (GCP) and Amazon Web Services (AWS).

The analysis shows a significant difference in the monthly costs between the two providers. GCP estimates the cost to be 163.20 dollars per month, while AWS estimates the cost to be 212.40 dollar per month for the same instance configuration. This difference of 49.20 dollars per month can add up quickly, especially if you're running multiple instances or scaling your infrastructure over time.

Furthermore, when evaluating cloud services, it's crucial to consider the broader ecosystem and the available services and tools that can integrate with your machine learning pipelines. Factors such as data storage, data processing, model training, and deployment capabilities, as well as the availability of pre-trained models, libraries, and frameworks, can all influence the overall cost and efficiency of your machine learning operations.

[1] https://cloud.google.com/
[2] https://aws.amazon.com/

Chapter 6

6 Discussion

In the context of MLOps (Machine Learning Operations), a RACI chart can be a useful tool to clearly define the roles and responsibilities of each team member throughout the end-to-end MLOps process. This process typically starts from the initial setup of the MLOps pipeline to the deployment of a production-ready machine learning model. The MLOps Development phase involves testing models in different environments, such as the local development environment or a dedicated server environment. The choice of environment often depends on factors like the size of the dataset, the computational resources required to train the model, and the overall scale of the project. For smaller-scale projects with moderate dataset sizes, development may be conducted entirely within the local development environment. This allows for faster iteration and easier debugging during the model development phase. However, as the complexity and scale of the project grow, it often becomes necessary to perform experiments in a dedicated server environment. This provides access to greater computational power and storage resources, enabling the team to handle larger datasets and more computationally intensive model training processes.

6.1 Development Discussion

We used different environments to develop the machine learning model based on the datasets. For the small dataset, we chose the local environment to develop the machine learning model. This was because the local environment is easy to manage and configure the machine learning model. Moreover, the local environment does not require a significant investment in resources, making it a cost-effective solution for developing the model.

The advantage of using the local machine is that we don't have to spend a lot of costs to develop the machine learning model. We can use the small resources available on the local machine to develop the model and prepare it for deployment. Additionally, for the small dataset, it is easy to manage and store the data on the local machine. Retrieving the dataset from different places and collecting it on the local machine is a straightforward process.

On the other hand, for the larger dataset, we opted to develop the machine learning model in a server environment. Specifically, we chose to use AWS Sagemaker for this purpose. Sagemaker provides a robust and scalable platform for developing and deploying machine learning models. The service offers a wide range of instance types, each with different capacities in terms of CPU and RAM, that can handle the processing of large datasets and complex machine learning algorithms.

The decision to use Sagemaker was based on the fact that we had a large amount of data to prepare for training the machine learning algorithm. Sagemaker's ability to provision instances with the necessary resources to interact with the dataset and the machine learning algorithm made it an ideal choice for this project.

6.2 Reflection

This thesis provides a well-researched and practical exploration of the challenges and best practices for implementing an MLOps workflow and defining roles in an industrial setting. It could serve as a valuable reference for organizations looking to industrialize their machine learning operations.

1. Comprehensive overview of MLOps and its integration with DevOps:

   - The paper provides a detailed background on the DevOps lifecycle and how DevOps principles can be applied to machine learning operations (MLOps).
   - It highlights the importance of end-to-end machine learning and the need for a structured MLOps workflow to manage the complexities of deploying and maintaining ML models in production.

2. Defining roles and responsibilities using RACI matrix:

   - A major focus of the paper is on defining the roles and responsibilities of different stakeholders (business, data, ML, DevOps) within the MLOps lifecycle.
   - The use of the RACI (Responsible, Accountable, Consulted, Informed) matrix is a structured approach to clearly assign the involvement of each role in the various MLOps activities.
   - This is a crucial aspect for coordinating cross-functional teams and ensuring smooth collaboration in an industrial MLOps implementation.

3. Practical industrial application:

   - Unlike many academic papers, this thesis specifically targets the implementation of MLOps in an industrial/enterprise setting.
   - The insights and frameworks presented are intended to be directly applicable and useful for organizations looking to adopt MLOps practices.
   - This focus on the industrial context makes the findings more relevant and impactful for real-world deployment of ML systems.

4. Thorough literature review and research methodology:

   - The paper demonstrates a comprehensive literature review of relevant sources on DevOps, MLOps, and RACI frameworks.
   - The research methodology appears to be structured, drawing insights from multiple papers and sources to build a cohesive understanding of the problem domain.

6.3 Future Work

As our organization continues to advance our MLOps practices, there are several areas we can explore to further improve the efficiency and traceability of our machine learning models. One key focus area is integrating MLFlow into our MLOps workflow. MLFlow is a powerful open-source platform for managing the end-to-end machine learning lifecycle, from experimentation to production deployment. By leveraging MLFlow, we can centralize the tracking of our model training runs, versions, metrics, and artifacts. This will provide much greater visibility and audibility into the model development process. Additionally, MLFlow's model registry and deployment capabilities will enable us to more seamlessly move models into production environments and monitor their performance. We can also explore MLFlow's experiment tracking to gain deeper insights into how model hyperparameters and configurations impact model performance. Integrating these MLFlow capabilities into our existing MLOps toolchain has the potential to drive significant efficiencies and enhance the overall robustness of our machine learning initiatives. As we continue to mature our MLOps practices, adopting MLFlow will be a key priority to optimize our model development lifecycle.

Chapter 7

7 Conclusion

This report presents a thorough exploration of the key components and processes involved in implementing an effective MLOps (Machine Learning Operations) workflow within an industrial setting. The central focus of the work is to define the roles and responsibilities of various stakeholders through the application of the RACI (Responsible, Accountable, Consulted, Informed) framework, as well as to design an overarching MLOps architecture to support the end-to-end machine learning lifecycle.

The report begins by providing a comprehensive literature review on the DevOps lifecycle and the principles of applying DevOps practices to machine learning projects. It then delves into the details of the End-to-End MLOps workflow, covering the crucial stages of business understanding, data acquisition and preparation, model development, model deployment, and model updating/retraining.

A key contribution of this work is the detailed RACI role definition for the MLOps lifecycle. The report clearly delineates the responsibilities of stakeholders such as business owners, data engineers, data scientists, software engineers, ML engineers, and DevOps engineers, ensuring seamless collaboration and accountability throughout the process.

Furthermore, the report presents the design of the MLOps architecture, highlighting the integration of various components like data pipelines, model training, model registry, model deployment, and monitoring. This holistic architectural approach enables the end-to-end automation and governance of the machine learning lifecycle, a crucial aspect of successful MLOps implementation.

The comprehensive nature of this report, covering both the conceptual and practical aspects of MLOps, makes it a valuable resource for organizations seeking to establish robust and scalable machine learning operations. The detailed workflow, role definitions, and architectural design provide a solid foundation for practitioners to adapt and implement MLOps within their own industrial settings.

In conclusion, this report serves as a comprehensive guide for organizations looking to bridge the gap between machine learning research and production-ready deployments. By effectively integrating DevOps principles and the RACI framework into the MLOps lifecycle, organizations can unlock the full potential of their machine learning investments and drive sustainable competitive advantages.

A Appendix

A.1 YAML script For CICD with GitHub Action

```yaml
name: workflow

on:
  push:
    branches:
      - main
    paths-ignore:
      - 'README.md'

permissions:
  id-token: write
  contents: read

jobs:
  integration:
    name: Continuous Integration
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v3

      - name: Lint code
        run: echo "Linting repository"

      - name: Run unit tests
        run: echo "Running unit tests"

  build-and-push-ecr-image:
    name: Continuous Delivery
    needs: integration
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v3

      - name: Install Utilities
        run: |
          sudo apt-get update
          sudo apt-get install -y jq unzip
      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${{ secrets.AWS_SECRET_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: ${{ secrets.AWS_REGION }}

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v1

      - name: Build, tag, and push image to Amazon ECR
        id: build-image
        env:
          ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
          ECR_REPOSITORY: ${{ secrets.ECR_REPOSITORY_NAME }}
          IMAGE_TAG: latest
```

```
        run: |
          # Build a docker container and
          # push it to ECR so that it can
          # be deployed to ECS.
          docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
          echo "::set-output name=image::$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG"


  Continuous-Deployment:
    needs: build-and-push-ecr-image
    runs-on: self-hosted
    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${{ secrets.AWS_SECRET_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: ${{ secrets.AWS_REGION }}

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v1


      - name: Pull latest images
        run: |
         docker pull ${{secrets.AWS_ECR_LOGIN_URL}}/${{ secrets.ECR_REPOSITORY_NAME

      # - name: Stop and remove  container if running
      #   run: |
      #    docker container stop stuperformance && docker container rm stuperformance

      - name: Run Docker Image to serve users
        run: |
         docker run -d -p 8080:8080 --ipc="host" --name=stuperformance -e 'AWS_ACCES
         docker system prune -f
```

A.2 Project Architecture Workflows

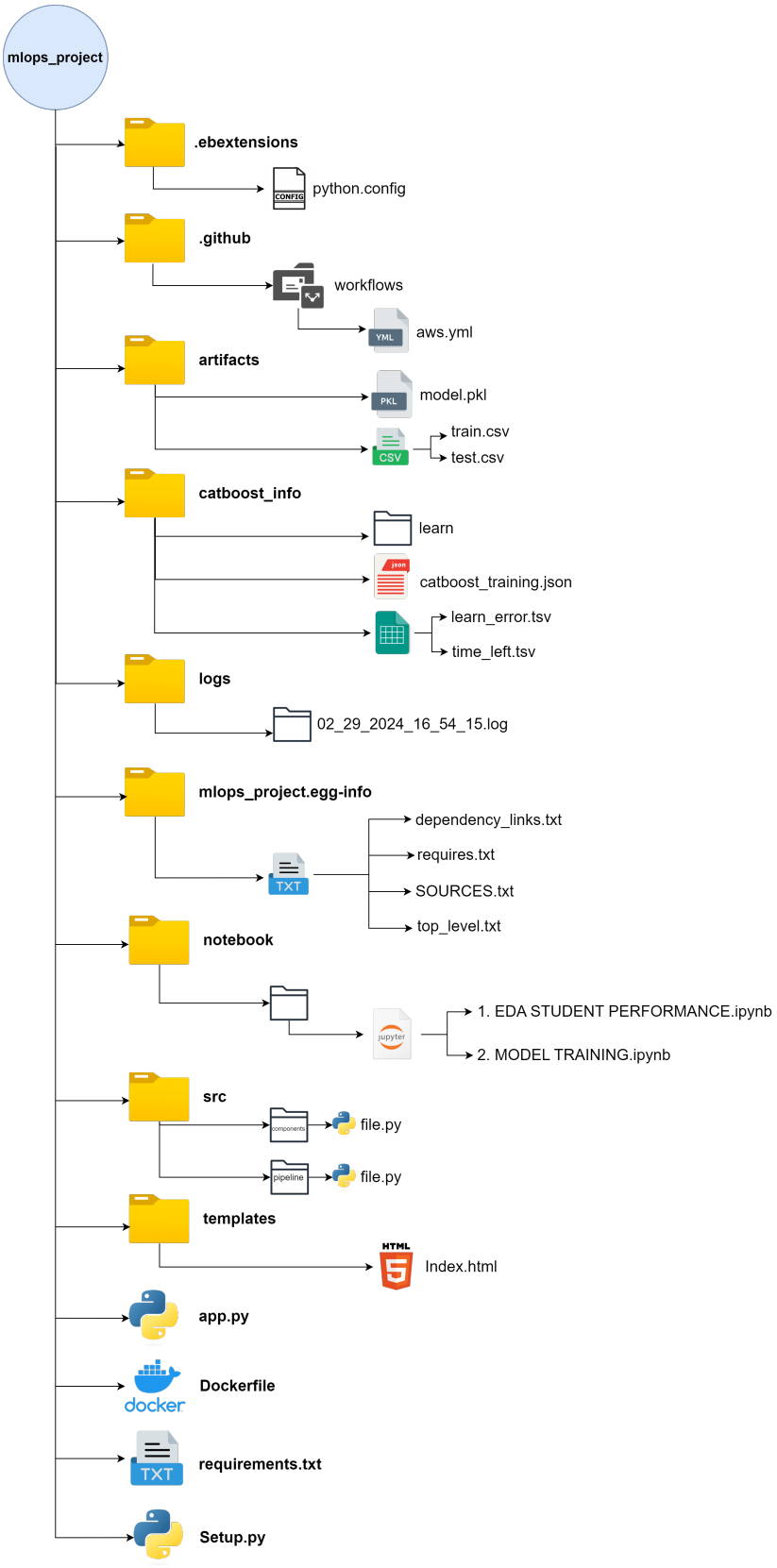

Figure 12: MLOps Project Architecture

References

[1] Laith Alzubaidi et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of big Data* 8 (2021), pp. 1–74.

[2] John T Behrens. "Principles and procedures of exploratory data analysis." In: *Psychological methods* 2.2 (1997), p. 131.

[3] Qifang Bi et al. "What is machine learning? A primer for the epidemiologist". In: *American journal of epidemiology* 188.12 (2019), pp. 2222–2239.

[4] Gerard Blokdijk. *The Service Level Agreement SLA Guide-SLA book, Templates for Service Level Management and Service Level Agreement Forms. Fast and Easy Way to Write your SLA*. Lulu. com, 2008.

[5] Kevin Brennan et al. *A Guide to the Business Analysis Body of Knowledger*. Iiba, 2009.

[6] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised learning". In: *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer, 2008, pp. 21–49.

[7] *DevOps Lifecycle : Different Phases in DevOps*. `https://www.browserstack.com/`. Feb. 24, 2023. URL: `https://browserstack.com/guide/devops-lifecycle#:~:text=The%20DevOps%20lifecycle%20is%20a,feedback%20throughout%20the%20software's%20lifecycle.`.

[8] Yanqing Duan, John S Edwards, and Yogesh K Dwivedi. "Artificial intelligence for decision making in the era of Big Data–evolution, challenges and research agenda". In: *International journal of information management* 48 (2019), pp. 63–71.

[9] Satvik Garg et al. "On continuous integration/continuous delivery for automated deployment of machine learning models using mlops". In: *2021 IEEE fourth international conference on artificial intelligence and knowledge engineering (AIKE)*. IEEE. 2021, pp. 25–28.

[10] Zoubin Ghahramani. "Unsupervised learning". In: *Summer school on machine learning*. Springer, 2003, pp. 72–112.

[11] Xinyi Du-Harpur et al. "What is AI? Applications of artificial intelligence to dermatology". In: *British Journal of Dermatology* 183.3 (Mar. 2020), pp. 423–430. DOI: `10.1111/bjd.18880`. URL: `https://doi.org/10.1111/bjd.18880`.

[12] Nipuni Hewage and Dulani Meedeniya. "Machine learning operations: A survey on MLOps tool support". In: *arXiv preprint arXiv:2202.10169* (2022).

[13] Ioannis Karamitsos, Saeed Albarhami, and Charalampos Apostolopoulos. "Applying DevOps practices of continuous automation for machine learning". In: *Information* 11.7 (2020), p. 363.

[14] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. "Machine Learning Operations (MLOps): Overview, Definition, and Architecture". In: *IEEE Access* 11 (2023), pp. 31866–31879. DOI: `10.1109/ACCESS.2023.3262138`.

[15] I Sessione di Laurea. "Mlops-standardizing the machine learning workflow". PhD thesis. University of Bologna Bologna, Italy, 2021.

[16] Matteo Migliorini et al. "Machine learning pipelines with modern big data tools for high energy physics". In: *Computing and Software for Big Science* 4.1 (2020), p. 8.

[17] *mlops-concepts*. URL: `https://app.datacamp.com/learn/courses/mlops-concepts`.

[18] Prajal Save et al. "A novel idea for credit card fraud detection using decision tree". In: *International Journal of Computer Applications* 161.13 (2017).

[19] Yan-Yan Song and LU Ying. "Decision tree methods: applications for classification and prediction". In: *Shanghai archives of psychiatry* 27.2 (2015), p. 130.

[20] T. J. (2019) Sturgeon. "Upgrading strategies for the digital economy". In: *Global Strategy* (2019).

[21] Suryakanthi Tangirala. "Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm". In: *International Journal of Advanced Computer Science and Applications* 11.2 (2020), pp. 612–619.

[22] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

[23] Venkatesh Upadrista. *Formula 4.0 for Digital Transformation: A Business-Driven Digital Transformation Framework for Industry 4.0*. CRC Press, 2021.

[24] Ravi Teja Yarlagadda. "DevOps and its practices". In: *International Journal of Creative Research Thoughts (IJCRT), ISSN* (2021), pp. 2320–2882.

[25] Yizhen Zhao. “MLOps Scaling ML in an Industrial Setting”. PhD thesis. PhD thesis, University of Amsterdam, 2021.