Building a generalisable ML pipeline at ING

by

Niels Bauman

to obtain the degree of Master of Science
in the Software Engineering Research Group
at the Delft University of Technology.

Student number:	4647165	
Project duration:	November 2021 – July 2022	
Thesis committee:	Prof. L. M. da Cruz,	TU Delft, Daily supervisor
	Prof. dr. A. van Deursen,	TU Delft, Thesis advisor
	Prof. J. Yang	TU Delft
Company supervisor:	P. Braakman	ING

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft ING

Abstract

Advances in data science have caused an increase in the use of Artificial Intelligence (AI), specifically Machine Learning (ML), throughout various fields. Not only in research but in the industry as well, has ML been receiving increasing amounts of interest. Many companies rely on ML models to increase the efficiency of existing processes or offer new services and products. The industry, however, is facing several additional challenges compared to the academic context. One of those challenges is applying the Development Operations (DevOps) model to an ML application, also referred to as MLOps. This thesis sets out to find the specific challenges that practitioners encounter while operationalising ML models. To do so, we perform a single-case case study on an ML pipeline built by the Trade & Communication Surveillance team at the ING bank. This case study consists of conducting a set of interviews and performing a manual code inspection of the pipeline. The team faces challenges ranging from having insufficient time for operationalising each ML project individually to operating in the highly-regulated fintech context. Their pipeline is able to deploy a single ML model but it does not generalise well to other projects. We present the first version of an application that mitigates these challenges. The application is able to deploy ML models to the development environment at ING and can be operated by data scientists to reduce the effort of operationalising an ML model.

Preface

This thesis was conducted at TU Delft's Software Engineering Research Group (SERG) under the AI for FinTech Research (AFR) Lab, which is a research collaboration between ING and TU Delft. ING is a bank with a global presence and a strong European base and offers retail and wholesale banking services to 38 million customers with over 57,000 employees [12]. The bank started as a traditional bank but is now at the forefront of the digital transformation. Since ING has been known for its technological advancements in the past few years, it presents itself as an IT company with a banking license. With the large amounts of data and varying products that ING offers, it provides extensive use-cases in applying AI. The AFR lab is a joint research effort to explore and dive into the possibilities of ML and AI in general within ING.

Throughout this thesis, I have been in contact with several employees at ING. In the initial case study phase, I mostly worked by myself but during the development phase, I regularly participated in the daily and weekly Scrum meetings with the team at ING. Besides those Scrum meetings, I held bi-weekly meetings with two members of the team to discuss my thesis progress and possible impediments.

I would like to thank my academic supervisor Luís Cruz for his support and feedback during this thesis. He always maintained the "glass half-full" perspective which really helped me push through more difficult times and decisions. Besides the interesting discussions we had on the thesis itself, we had a lot of fun in my experience. Without him, I would definitely not have enjoyed my thesis as much as I did. My thanks also go out to Prof. Arie van Deursen for being my thesis advisor and providing additional feedback from an academic point of view. Finally, I thank Jie Yang for being part of my thesis committee.

On the ING side, I would like to thank Paul Braakman, who is the chapter lead of the Trade & Communication Surveillance squad, for his extensive support throughout my thesis. Besides being part of the stakeholders for the application, he contributed his insights from an academic perspective as well. We always had a great time during meetings and both of us would get really enthusiastic when we talked about the possibilities of the application. I would also like to thank Xuehan Jiang for his feedback and support throughout my thesis. He provided useful insights into his day-to-day workings as a data scientist at ING. My thanks also go out to all the members of the Trade & Communication Surveillance squad and the AFR lab.

Niels Bauman
Delft, June 2022

Contents

1 Introduction

Advances in data science have caused an increase in the use of Artificial Intelligence (AI), specifically Machine Learning (ML), throughout various fields [7, 16, 6]. Not only in research but in the industry as well, has ML been receiving increasing amounts of interest [14, 10]. Many companies rely on ML models to increase the efficiency of existing processes [21] or offer new services and products [9]. Davenport describes how companies can achieve a substantial competitive advantage by leveraging data sources through analytical tools [8].

The industry, however, is facing several additional challenges compared to the academic context. One of those challenges is applying the Development Operations (DevOps) model to an ML application, also referred to as MLOps. For traditional software applications, DevOps means combining developmental and operational tasks within one workflow. For MLOps, several additional challenges make it more complex than traditional DevOps [3]. ML models often require large amounts of data to optimise performance. This data needs to be stored, managed and consumed, which poses a common challenge in ML projects. Another complex MLOps-specific challenge is setting up a so-called *feedback loop*. The purpose of a feedback loop is to improve the performance of the model while it is being consumed already. The loop feeds validated scoring data back into the training stage of the model.

These challenges can hinder practitioners in operationalising ML models. This, in turn, ”can easily jeopardise the success of an entire project” [3]. This means that failing to overcome MLOps challenges results in ML projects being unable to be utilised in actual production systems. This would be a huge waste of the time and effort that were already put into the projects. It is therefore of great significance to further analyse these challenges and experiment with solutions to overcome them.

With the large amounts of data that ING has, it has the perfect basis for developing ML models. People often think of using customer data for ML applications. However, the team that is being examined in this study, is concerned with Trade & Communication Surveillance. This team is tasked with maintaining the first and second lines of defence regarding monitoring trade and communication within the bank. They aim to flag suspicious behaviour such as insider trading and other forms of fraud. Their current setup consists of a system that automatically flags emails that contain some form of fraud. This system currently uses the following process: it simply runs the emails by a list of keywords and if there is a match, it will flag them. This process identifies numerous messages as potentially indicating suspicious behaviour. However, most of them are false positives: during the period between January - August 2021 a total of 157k false positives were reported. These flagged emails are then manually checked by a team to filter all the false positives. According to the team, a minimum of 45 seconds is required to manually process a single alert.

After performing some statistical analysis on the reported alerts, the team discovered that during that same period 47% of all alerts were so-called *mailing lists*. A mailing list is simply an email that is sent to a collection of people at the same time. In the team’s experience, this often means the email will not contain the type of fraud they are surveying. This is the reason why they invested time in building an ML model that aims to predict whether an email can be classified as a mailing list. While this model might not directly be able to disregard alerts, it would, as a first step, be able to aid the manual checkers in their decision. The team managed to implement this ML model and built a pipeline around it that was able to deploy the model.

This mailing list model was just one of many ideas for ML models they had to improve the process of reporting fraudulent emails. For instance, they had also already started developing a model that would be able to classify a paragraph as a disclaimer. This could be used to filter certain paragraphs from flagged emails and thereby reducing manual work. While the team had already set up a pipeline for the mailing list model, they wanted to avoid simply copying that code to this project. Therefore, they were searching for a solution that would avoid this duplication. Since these ML projects were not task obligated by regulations but rather of innovative nature, only limited time would be available. This required the application to be low in maintenance with a minimal learning curve as well.

1.1. Research Questions

This thesis sets out to answer the following research questions:

RQ1 What are the challenges of operationalising ML models?

RQ2 What does an industry solution to operationalising ML models look like?

RQ3 Can we have a deployment solution for ML models in a highly-regulated context that can be operated by data scientists?

We start by performing a case study on the Trade & Communication Surveillance team to answer **RQ1**. By conducting interviews with members of the team, we can identify the challenges that this team experienced while operationalising their ML model. Some of these challenges are operational, such as prioritising innovation to educating team members on MLOps practice. Others are more practical, such as dealing with confidential data in the highly-regulated fintech sector.

To answer **RQ2**, we analyse and document an industry-built ML pipeline. We inspect the pipeline code and its design in general. The pipeline is able to deploy the model but it does not generalise well to other projects.

Finally, we build a custom application that is able to deploy various ML models, as an answer to **RQ3**. This application serves as a first version and lays the groundwork for various features.

1.2. Contributions

This thesis contributes to existing research by documenting an ML pipeline that is used in practice. Due to the infancy of MLOps, it is valuable to document as many practical examples as possible. This provides examples to experts and other researchers, who can learn and extend these works. One of the main findings of this study is that while MLOps on its own poses a challenge, applying MLOps in a highly regulated sector such as fintech complicates the process even more. An example of such a fintech-specific challenge is the absence of tools for managing ML model workflows within such a highly-regulated context. Finally, due to the limitations of these available tools, a custom solution is required to be able to operationalise ML models. We present an initial version of such a custom solution.

1.3. Thesis Outline

This thesis starts by discussing related work in Chapter 2. Chapter 3 describes the case study on the team at ING. In Chapter 4, we describe the solution application that aims to mitigate the challenges found in the case study. Finally, Chapter 5 concludes the thesis and makes suggestions for future work.

2

Related work

Even though MLOps is still a relatively young field of research, there have already been several case and empirical studies performed. This chapter will take a look at some of the existing work on MLOps specifically and ML development in general.

Zhou et al. [25] describe their case study of an ML pipeline platform. Whereas this thesis looks at an existing pipeline, they have specifically created a pipeline from scratch for their study. They do not consider a custom ML model. Instead, they use several approved models such as GoogleLeNet. Their setup employs the tools Kubeflow, Gitea and Drone. The focus of their study mainly lies on the time and resource consumption of certain tools and stages of the model lifecycle. While these aspects are of great significance for ML projects, they lie outside of the scope of this thesis.

John et al. [13] present an MLOps framework that details the activities involved in the continuous development of ML models. They do so by first performing a literature review on MLOps. Secondly, they present a maturity model in which they outline the different stages that companies go through in evolving their MLOps practices. Finally, they validate their framework in three case companies and apply their maturity model to those companies.

Tamburri [22] describes his research into trends and challenges regarding sustainable MLOps. He argues that the more MLOps platforms penetrate the day-to-day activities of software operations, the more AI Software will face the risk of becoming unsustainable from a social, technical or organisational perspective. By analysing the operations of his research institution, the author extracts several challenges. The challenges he identifies mainly originate from the operational side of MLOps and can be categorised into the following four categories: explanation, fairness, accountability and sustainability.

Ruf et al. [20] present their study on MLOps tools. They examine a set of tools available for MLOps workflows and compare these tools to one another. The main focus during these comparisons is the inter-connectivity of specific tools. They draw the conclusion that no single tool has the capability of realising a fully automated MLOps workflow. They also conclude that different tools have overlapping features, which should be taken into consideration during the initial setup of future projects. Several of these tools will be further discussed in Section 4.2.

Arpteg et al. [2] perform a set of seven case studies to identify software engineering challenges in ML, specifically Deep Learning (DL). The seven projects that they analyse, are from companies that range from start-up size to large multinational companies. In their study, they identify 12 challenges that can be grouped into developmental, productional and organisational challenges. The main focus of the paper is not to provide solutions, but rather to outline problem areas. They conclude that there is still a significant need for further research into how to easily and efficiently build production-ready ML systems.

Breck et al. [5] express the importance of testing and monitoring in real-world production ML systems. Based on the years of experience of using ML at Google, they have developed a set of best practices for using ML systems. Their rubric covers a range from a team just starting out with ML up to tests that even a well-established team may find difficult. Similar to previous studies, this study focuses mainly on the operational aspects of MLOps.

Both Paleyes et al. [18] and Baier et al. [3] describe their surveys of case studies on challenges in deploying ML applications. They perform literature reviews on case studies and reports to extract prob-

lems and concerns practitioners face. Both studies discuss various challenges, both developmental and operational. Paleyes et al. also briefly discuss some potential solutions to these issues. They mention several commonly used tools but don't go into further detail on them. Finally, both conclude that a poor deployment experience can severely hinder the further growth of ML adoption. While both studies perform research on MLOps challenges, neither address the additional challenges in a highly-regulated setting.

Ananth et al. [1] present their white paper on the deployment of AI models in banking. They touch upon both developmental and operational challenges, some specific to the banking context. One of their main banking-specific challenges is dealing with strict regulations. These strict regulations are, for instance, closely related to the explainability of AI, which is a whole sub-research field of its own. While this thesis similarly focuses on the banking context, it additionally focuses on the challenges of transdisciplinary teams. The application presented in this thesis is specifically designed to make sure non-technical stakeholders are also part of the operations.

A variety of case studies has been performed, but very few also describe the actual solutions. This prevents both academics and practitioners from learning from in-use systems. While a single system might not yet be the perfect solution, documenting designs and the reasoning behind them will help experts iterate over solutions and build new and improved solutions. The studies that do present the design of an MLOps solution tend to focus on deploying a single ML project. While this approach works on an individual project basis, these solutions do not generalise well to multiple projects as they require extensive configuration. This thesis aims to prevent that limitation by presenting an application that does generalise well to multiple projects.

3

Case study

To identify the challenges of operationalizing ML models and to analyse an industry solution, we perform a single-case case study. By looking at a concrete situation, we can define new research and shed empirical light on existing concepts and principles [23]. This chapter starts by describing the methodology for this case study. It follows by presenting the results of this study. Finally, the chapter discusses these results.

3.1. Methodology

This case study closely investigates an industry-built ML pipeline. Although the pipeline is not yet being utilised in a production environment at the time of writing, analysing the approach will bring useful insights into some of the challenges that MLOps faces. Besides an in-depth analysis of the pipeline code, interviews have been conducted with some of the engineers that actually developed the pipeline, as well as members of the same team that did not contribute to this pipeline.

3.1.1. Interviews

The main data source for this study is a set of interviews. These interviews were semi-structured; there was no predefined set of questions for each interview, but rather a flowing discussion was held. The interviews took approximately one hour and were conducted with the help of another researcher to prevent cognitive bias. The approach used to extract and collect information from the interviews is based on the guidelines proposed by Halcomb and Davidson [11]. The process is as follows:

1. Audio taping of the interview and concurrent note-taking.
2. Reflective journaling immediately post-interview.
3. Listening to the audiotape and revising notes.
4. Content analysis.

Participants

We selected five participants, based on their roles, for the interviews. All the participants were members of the Trade & Communication Surveillance team at ING. By including participants with varying roles, we were able to obtain different perspectives on the subject. In total, 5 participants were interviewed. An overview of the participants can be seen in Table 3.1.

3.1.2. Manual Code Inspection

Besides the interviews, we performed a manual code inspection of the ML pipeline that was implemented by the team. The inspection was intended to confirm the results of the interviewers and possibly provide additional insights as well. In the inspection, both the code for the ML model itself and the CI/CD setup was considered, to get a good grasp on all MLOps aspects of the pipeline.

ID	Role
P1	Dev Engineer
P2	Compliance Specialist
P3	Data Scientist
P4	Chapter Lead
P5	IT Risk Manager

Table 3.1: Overview of the participants.

3.2. Results

After conducting the interviews, we analysed the notes and extracted several concepts by grouping related notes together. These concepts are discussed in the following subsections. While discussing a concept, we will note the participants' IDs that talked about that concept. Finally, Section 3.2 describes the pipeline itself in detail.

Motivation

One of the first topics discussed in the interviews was the motivation for concentrating efforts on building an ML pipeline. All of P1, P2 and P4 mentioned the desire for automation as one of the key motivations for developing the initial ML model for this pipeline. The motivation for building the pipeline itself arose from the aspirations to build additional models. The idea of the pipeline was that models could be plugged in with little effort and would then be automatically deployed to various environments to finally be used in production. This would reduce the manual work associated with deploying an ML model and would thus improve the speed at which ML models can be developed.

Even though there was enough motivation for building the ML models and the pipeline, something that P1, P2 and P4 talked about was the challenge of prioritising innovation. Also for companies that strive for innovation, like ING, there are always certain tasks that will have to be completed and deadlines to be met. This makes it difficult to prioritise projects that are not ”required”; they might probably only provide benefit in the long run.

This team specifically was also dealing with developer retention during the development process of the pipeline, which both P1 and P4 described. This of course makes prioritising innovation even more difficult. Unfortunately, this is also not something for which a lot can be done in the short run and is still difficult to manage in the long run.

For most ML projects one of the hardest challenges is collecting and cleaning data. Fortunately, this team had the advantage that they were already in possession of large amounts of data, as confirmed by both P1 and P4. This data was also easily accessible through an API. While this data was also in fact labelled, it was labelled by humans, which means there is human error possible in those labels. Considering the amount of available data, however, this did not pose a significant problem.

Regulations

As a bank, ING has to obey numerous regulations set by organisations all over the world. These regulations influence the way developers at ING have to approach developmental tasks in general, but even more so for ML tasks since those rely heavily on data. At the time of writing, there are no regulations set for the Trade & Communication Surveillance domain within the bank regarding ML applications specifically. This meant that the team had to step into the role of the regulator while developing the ML models, as mentioned by P2 and P4. By looking at existing non-ML regulations as well as envisioning strict rules during their development, the team aimed to make the model and pipeline ready for future regulations.

To comply with all regulations, the team was forced to partially implement the pipeline on a Virtual Private Cloud (VPC), hosted by ING. By using this VPC, all sensitive data is only stored and processed on servers that are managed by ING, meaning that the risk of a leak or breach is far lower than on public servers. Fortunately, the Continuous Integration/Continuous Deployment (CI/CD) part of the pipeline was allowed to run on public servers, meaning that the team was able to use existing tools for the CI/CD.

Because the team maintained and envisioned strict rules for both the models and the pipeline, they included a manual step in the pipeline. This was mentioned by P2, P4 and P5. Although this limits the

extent to which the pipeline can be automated, the team felt it was required to always be able to ensure a certain performance of the pipeline.

Collaboration
A recurring topic during the interviews was the collaboration on ML projects. In traditional software projects, there could already exist a slight knowledge gap between software engineers and DevOps experts. For ML projects, however, the gap is even larger as most ML experts are not familiar with standard DevOps practices. Analogously, DevOps experts are generally not familiar with ML practices. P1, P3, P4 and P5 all mentioned the value of educating both sides with some of the basics of the fields.

With collaboration also comes responsibility. During the interviews, both P3 and P5 talk about managing responsibilities in an ML pipeline project. In the initial phases of developing an ML model, it is clear that the responsibilities lie with the ML experts. Once the model is part of a deployment pipeline, however, one person might still hold ML experts accountable for any sorts of unexpected results but another might hold the DevOps experts accountable. P5 advocates that the entire team should be responsible for the deployment of the model, which also highlights the value of educating both sides.

The Pipeline
The pipeline makes use of MLflow [17] to define the different stages of the model pipeline. These stages are: fetching data, cleaning data, training model, scoring data and pushing scores. While those stages can be called individually from the command line, the pipeline uses the approach suggested by one of the official MLflow tutorials for orchestrating multi-step workflows[1]. This approach defines one main python file that executes the right stages in the right order, depending on whether the user wants to train or score the model. Additionally, it checks with the running MLflow instance whether a particular stage has already been executed and if so, it will reuse its results. This prevents the pipeline from doing unnecessary duplicate work. An overview of the pipeline can be found in Figure 3.1.

MLflow has built-in support for the Python package manager Conda[2]. This allows the pipeline to easily define dependencies for the project and package the dependencies as one environment for deployment.

The CI/CD pipeline consists of several stages for Azure DevOps[3]: build, deploy, train and score. Figure 3.2 displays an overview of the stages and what they consist of. The build stage verifies that the model is still runnable without any errors. The deploy stage actually uploads the model to a Linux server and starts the MLflow tracking server. The training stage retrains the model on the Linux server. Finally, the scoring stage re-scores the data with the updated model. Each of the stages includes a ”prepare environment” step, which ranges from collecting secrets to downloading packages. The tasks that require connecting to the Linux server are performed by using Ansible, which is an open-source deployment tool.

3.3. Discussion
This section discusses the results obtained from the interviews and the manual code inspection.

Challenges
Most of the results obtained by the interviews are not unique to the banking domain. For a large part, the challenges related to motivation and collaboration can also be found in related works that do not focus on banking specifically. While strict regulations are not unique to banks either, they are only rarely discussed in any of the related works (cf. Chapter 2). The interviews did not go into detail on the regulations themselves, but banks will likely have to adhere to extremely strict regulations, often perhaps even more so than other types of organisations. It is interesting to note that this team currently has no regulations or guidelines to follow for their tasks. While ML and AI in general have been getting increasing attention in recent years, apparently some fields have yet to keep up with these new technologies.

We can now answer **RQ1** What are the challenges of operationalising ML models? Some of these challenges are organisational, such as prioritising innovation and working on ML projects with transdisciplinary teams. Others are more practical, such as operating in the highly-regulated fintech context.

[1] https://github.com/mlflow/mlflow/tree/master/examples/multistep_workflow
[2] https://docs.conda.io/en/latest/
[3] https://azure.microsoft.com/en-us/services/devops/

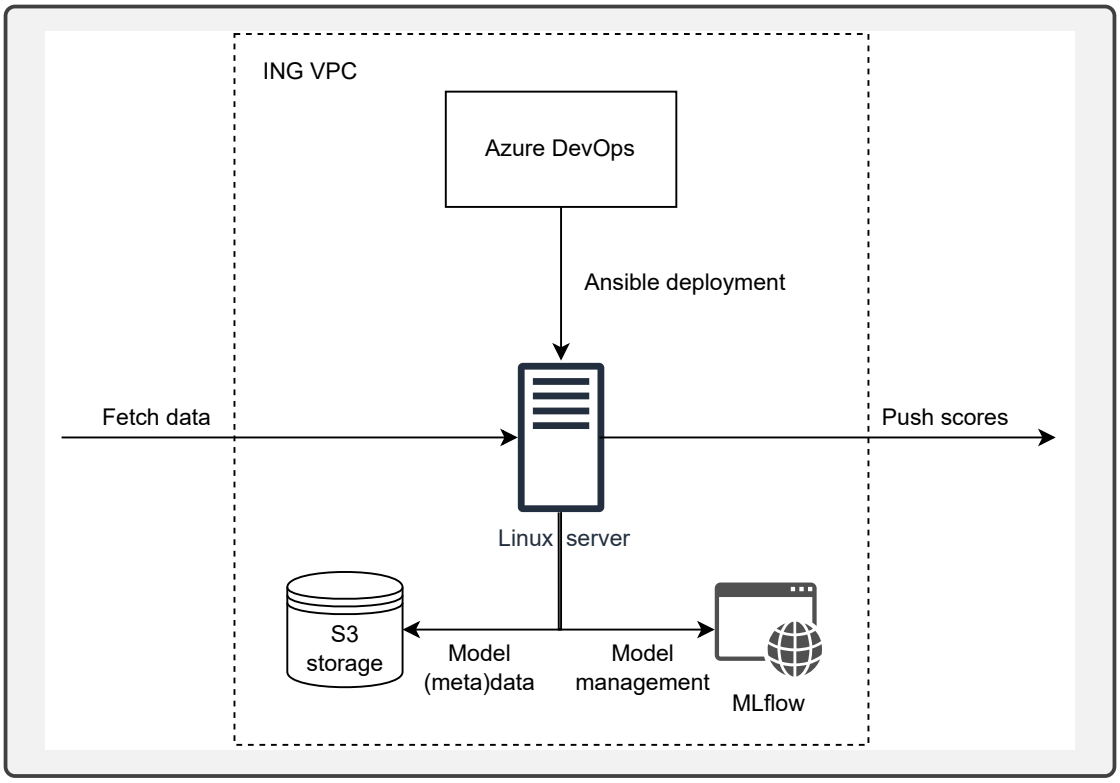

Figure 3.1: An overview of the setup of the pipeline.

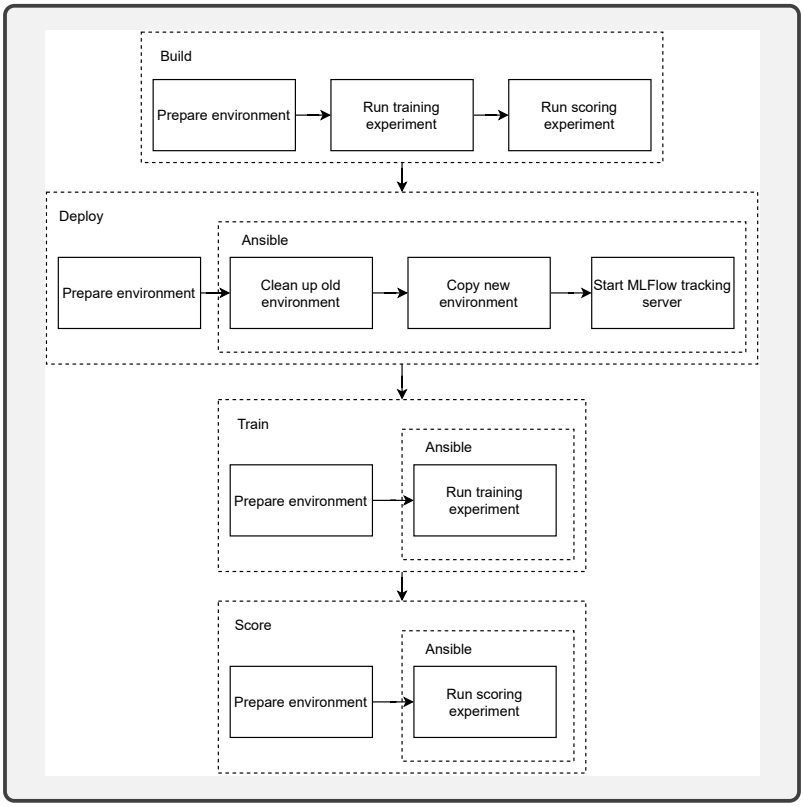

Figure 3.2: A flow diagram of the CI/CD part of the pipeline.

The Pipeline
The first remarkable aspect of the pipeline is the stage-invocation structure. Apparently, MLflow promotes the use of one main Python file that manually checks cached runs and invokes stages of the model pipeline. It is interesting that MLflow recognises the need for multi-stage model pipelines, but has not incorporated functionality that avoids having to duplicate boilerplate code. Also, while their proposed solution allows ignoring cached runs, it only does so for the entire run, i.e. you are not able to invalidate the cache of a single stage. This limits the development process as, for instance, a developer might only alter the training stage but will then have to wait for the data collection stage to complete even though that will simply return the same result as the cached run. MLOps frameworks ought to be designed in a way that promotes best practices and prevents technical debt.

Another notable aspect of the pipeline is the absence of a performance check before deployment. While the pipeline makes sure that the model can be trained successfully in the build stage, there is no validation of the performance of the model present. This means that a change of the model could be detrimental to the performance of the model but it would be deployed nonetheless. Once someone notices the decrease in performance, they would have to manually revert the change. A simple performance validation in the pipeline could prevent this performance drop and manual work. MLOps frameworks have an important opportunity to enable more efficient development approaches.

Next to the absence of a pre-deployment performance check, the pipeline is also missing a stage that performs data quality testing. With large amounts of data, it is important to regularly assess the quality of the data that is being processed by an ML model.

Even though the pipeline makes use of an API to retrieve data and push scores, this API does not support data versioning. This makes it difficult to track down which version of the model pushed a certain score of a data entry. Also, it makes it difficult to figure out which data exactly was used for training a certain version of the model.

The pipeline currently does not generalise or extend well to other models. There is some configurability present for the current model, but not enough to actually run another model alongside the current one. Fortunately, there are aspirations to revise the pipeline to make it more extendable for other models and possibly even more generalised for other teams as well.

We can now answer **RQ2** What does an industry solution to operationalising ML models look like? The solution built by the team at ING is able to deploy a single model to IPC by using the Azure DevOps CI/CD and makes use of MLFlow for model management.

3.4. Threats to Validity

The main limitation in this case study is the number of interviews held. Although we noticed some recurring concepts in the five interviews, having more interviews would have strengthened the results and possibly added additional conclusions as well. The size of the team that was investigated was rather small, so not a lot of participants could have been selected from that team. However, by scouting participants from other teams, we would have managed to achieve a higher number of participants. We chose to limit the scope of this study to this one particular team due to the following reasons. The main reason was that part of this thesis was to provide a solution that would work for this particular team. Hence, by including participants from other teams there would be first- and second-class participants which could change the overarching goal of the thesis. Since this thesis has a predefined duration, the available time was limited which was also a factor in limiting the scope. The procedure of this study is reported in a way that promotes reproducibility. Therefore, this work paves the way for a solution that generalises to other teams as well.

4

Metis

Now that we have established the challenges that this team at ING experienced whilst operationalising an ML model, we will try to mitigate these challenges. To do so, we will build an application called *Metis*. This name was chosen by the team at ING and originates from the Greek myth Metis which refers to a creature that represents ”wisdom” or ”skill”. This chapter describes the application in detail. It will start by listing all the requirements and limitations that were imposed on the application. This is followed by a thorough analysis of available MLOps tools. Afterwards, the development process is described, followed by the application design. The following section presents the evaluation methodology and results. Finally, these evaluation results are discussed.

4.1. Requirements

Before diving into the development phase, we first collected all the requirements from the stakeholders. The stakeholders consisted of the Trade & Communication Surveillance team at ING. These requirements can be split up into functional and non-functional requirements.

4.1.1. Requirements Collection

To meet the stakeholders' wishes as best as possible, several brainstorming sessions were held with several members of the team at ING together with the help of another researcher. The first brainstorming session took place online was held with two members of the team. To facilitate the brainstorming, the session made use of Google Jamboard[1]. The session was structured as follows:

1. **Introduction**: The session started with a general introduction. This introduction included the session agenda and the goals of the session. The goals of the session were to collect the team's wishes for the application and to think about possible solutions.
2. **Problem description**: Next came a short problem description. This made sure all participants were on the same page regarding the problem at hand. The problem was formulated as ”what does a generalisable ML pipeline look like”.
3. **Word web**: The first task of the session consisted of creating a word web. All participants were told to try and come up with as many features as possible that would fit an ML pipeline. Everyone used a unique colour for their ideas so it was easy to identify from whom the idea originated. To give the participants the freedom to think, everyone muted their microphone. After approximately 10 minutes everyone reconvened to continue the session.
4. **Proposing solutions**: Proposing solutions was the final task of the session. During this task, the participants held an open discussion on different possible approaches.

Figure 4.1 shows the final word web. Since all the entries were scattered around the Jamboard page, the participants grouped the entries on similarity. A total of 10 clusters can be extracted from the word web:

[1] `https://workspace.google.com/products/jamboard/`

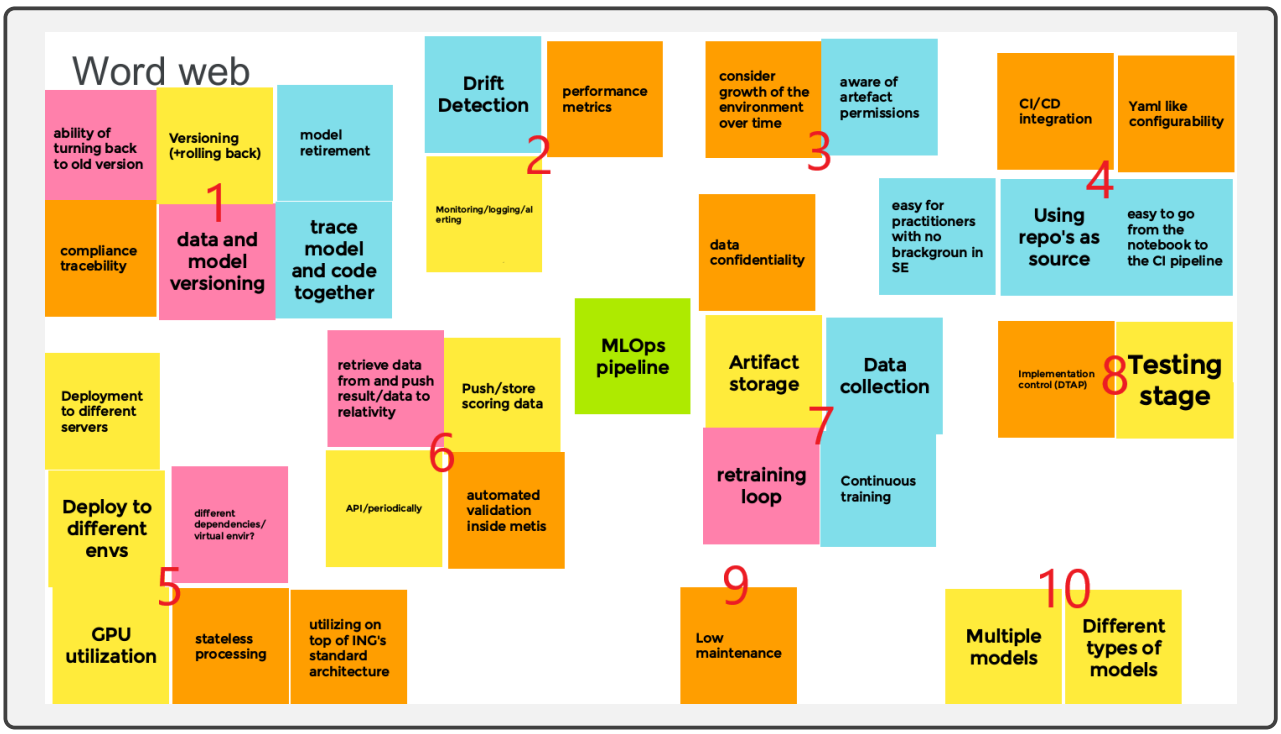

Figure 4.1: The word web from the brainstorming session.

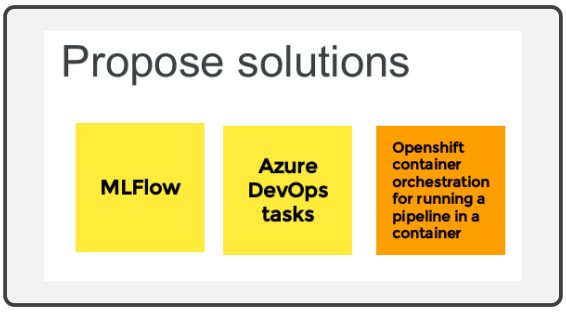

Figure 4.2: The solutions from the brainstorming session.

1. Version
2. Monitoring
3. Access control
4. Invocation/configuration
5. Hosting
6. Data retrieval
7. Data storage
8. Staging
9. Maintainability
10. Generalisability

The final discussion boiled down to the three main solutions depicted in Figure 4.2. Firstly, MLFlow was mentioned not as a full solution, but as a possible solution for a subset of the features such as versioning and monitoring. Secondly, the idea was proposed to implement a custom Azure DevOps task[2] that would deploy the model. Finally, RedHat's OpenShift container orchestration platform was suggested as a way to run a pipeline in containers.

After the initial brainstorming session, several unstructured discussions took place to further refine the application design. In the interest of time, we scoped this study to build a first version with a subset of the features. Together with the results of the case study, we formulated a set of requirements.

4.1.2. Functional

The main purpose of the application is to deploy ML models. The goal is to make this a straightforward process that requires little manual work. As can be traced back to Section 3.2, the team's original idea was to build a pipeline in which models can be plugged in with little effort. This would reduce the manual work associated with deploying an ML model and would thus improve the speed at which ML models can be developed.

Section 3.2 discusses the knowledge gap between data scientists and operations engineers. By making sure the steps required to set up a new project for this application are doable by data scientists, we can further improve the speed at which ML models can be developed. This will allow data scientists to deploy their models to remote environments themselves.

4.1.3. Non-functional

Some of the non-functional requirements do not originate from the team itself but are set by ING for all teams. One of the main ING-wide requirements is that the application is fully hosted within ING VPC. The reason for this requirement is that these models will be consuming highly confidential data. This data is under no circumstances allowed to leave the ING network. This is also why all servers hosted in IPC are, by default, not allowed to access the internet. Only through rigorous procedures is it possible to allow a server to connect to specific parts of the internet. The inability to access the internet directly poses a significant limitation on the application: it is not possible for the application to download project dependencies dynamically.

One of the non-functional requirements from the team itself is the programming language of the application. They require that the application is written in Python, version 3.8 or higher. Since almost all ML projects are written in Python, it makes sense to write the deployment application in the same language.

Within ING, there are four environment levels that an application can be deployed: Development, Testing, Acceptance and Production (DTAP). Each level comes with increasing amounts of regulations and requirements. In the interest of time, this study focuses only on deploying the application to the development environment for now. The team has access to several servers in these environments. Each of these services is running version 7.9 of the Red Hat Enterprise Linux (RHEL) OS.

4.2. Available Tools

There are already several well-known tools out there that aid both academics and practitioners in developing ML applications. Examples of those tools are Keras, scikit-learn and Tensorflow. Within the

[2] https://docs.microsoft.com/en-us/azure/devops/pipelines/process/tasks

field of MLOps, there are also several open-source tools available already, where some of which were developed by well-known teams like Google.

Tensorflow Extended (TFX) [4] is an ML platform based on Tensorflow. It provides a configuration framework to express ML pipelines consisting of TFX components. TFX pipelines can be orchestrated using other third-party tools such as Kubeflow pipelines. Both the components themselves as well as the integrations with orchestration systems can be extended. TFX is an end-to-end platform for deploying production-ready ML pipelines. It has relatively high integration costs which makes it less ideal for smaller projects. Additionally, it requires the use of the Tensorflow (TF) framework libraries to make use of TFX.

Kubeflow [15] is a specialised toolkit for orchestrating ML workflows. The goal of the tool is to make deployments of ML workflows simple, portable and scalable on Kubernetes. This dependency on Kubernetes requires a detailed configuration before being able to actually deploy a model. This in turn makes Kubeflow also less ideal for smaller projects.

MLflow [17] is an ML toolkit that focuses mainly on model and lifecycle management. It allows users to register and track models and model runs. This tracking includes reporting custom metrics, viewing logs and managing the model's artifacts. While it requires less effort than the other discussed tools to configure a basic setup with MLflow, the configuration is very specific to the project and thus does not generalise well to other projects. This means manual work is required for each new project. A large part of this manual work consists of repeating the same steps and should thus be automated.

ZenML [24] is a lightweight tool for creating ML pipelines. It provides a UI for visualising pipelines and comparing pipelines from different projects in one ZenML repository. It uses Apache Beam[3] to allow the user to construct a pipeline through a combination of steps. However, it does not contain built-in support for deploying models to remote environments. Instead, it relies on tools like Kubeflow to do so. Since Kubeflow itself is not a suitable solution for this study, ZenML is also not.

All these tools have in common that they require the user to intertwine ML code with "Operations" code. To use the tools, you will need to configure them using both Python and non-Python files, which makes projects more cluttered.

These tools have been developed to be as generalizable as possible. This makes sense since the tools want to be able to support as many projects as possible, but this means they can only assume very little about the project itself. For the case of this team at ING, we still want to make as few assumptions as possible, but we do have the ability to set some requirements if the benefits outweigh the consequences. Additionally, we have some foreknowledge on the possibilities of the available infrastructure. We can use this knowledge to make the application fit as best as possible to the available infrastructure.

A team within ING has been working on an MLOps application called the Machine Learning Platform (MLP) which is currently only available for internal use. It is solely designed for hosting ML models and therefore requires users to train their models beforehand. This still leaves users with the task to set up some form of pipeline that is able to run the necessary commands to train the model. Since these commands generally cannot be executed on the Azure DevOps CI/CD servers due to security measures at ING, a user would have to build some form of system that allows running these commands within IPC. This still leaves the user with substantial manual work and is thus not a suitable solution for this study.

One of the most significant limitations is the prohibition of internet access. While most tools can also be installed manually on servers without internet access, none of them has built-in support for mitigating this limitation when it comes to downloading the projects' dependencies. A user could mitigate this manually by creating custom docker images which contain the dependencies. However, this is a cumbersome process with several manual tasks and is therefore not a suitable solution for this study.

4.3. Development Process

Since none of the available open-source tools is a suitable solution, we chose to build a custom application. To avoid the common pitfalls of the waterfall model [19], we implemented an Agile development method. This method consisted of notifying the stakeholders twice a week through chat. These chat messages contained straightforward updates on the development progress. Additionally, bi-weekly meetings were held with the stakeholders. During these meetings, the progress of the application

[3] https://beam.apache.org/

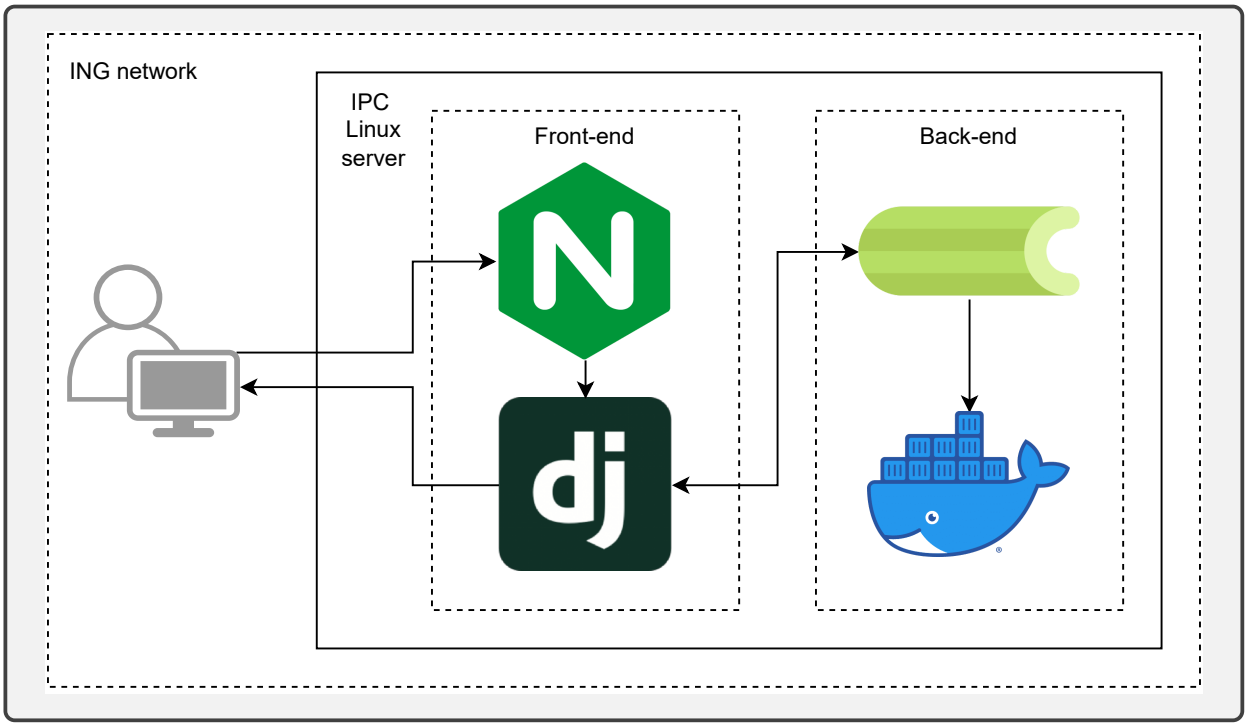

Figure 4.3: An overview of the setup of the pipeline.

would be presented and discussed. Often, these meetings also included a demo which gave the stakeholders clear insights into the progress of the application. The application was designed iteratively as well. The next section describes the final design.

4.4. Application Design

An overview of the final design is depicted in Figure 4.3. The application is fully hosted in IPC. This means that users are only able to connect to the application when they are connected to the ING network, through ING's VPN for example. Although this is obligatory by ING, it also provides an additional layer of security around the application. The application can be split up into two components: the front- and back-end.

4.4.1. Front-end

The front-end is the component that the user interacts with. It consists of an NGINX[4] server that serves a Django[5] application. The Django application contains both a User Interface (UI) and an Application Programming Interface (API). While users will often only interact with the UI, they are able to use the API directly as well.

The UI can be used to manage and upload models. To upload a model, a project has to be created first. Projects only consist of a name and the list of runs that have been uploaded for that project. Figure 4.4 shows a screenshot of the dashboard page where users can create a project. When a model is uploaded by the user, the API stores the uploaded content on disk and queues a background process to execute the model. Figure 4.5 shows a screenshot of a project-specific page. The *Status* section shows some general information on the most recently completed run. Next to that is the *Upload* section. This section is used to upload models. The UI also contains a help page that provides useful information to aid users in using the application. Finally, Figure 4.6 shows a screenshot of a run-specific page after the run has been completed. The top left *Status* section contains similar information as the project-specific page. Additionally, it includes a button to copy a cURL command to the user's clipboard to invoke the model, which will be described in more detail in Section 4.4.2. The *Manage*

[4] https://www.nginx.com/
[5] https://www.djangoproject.com/

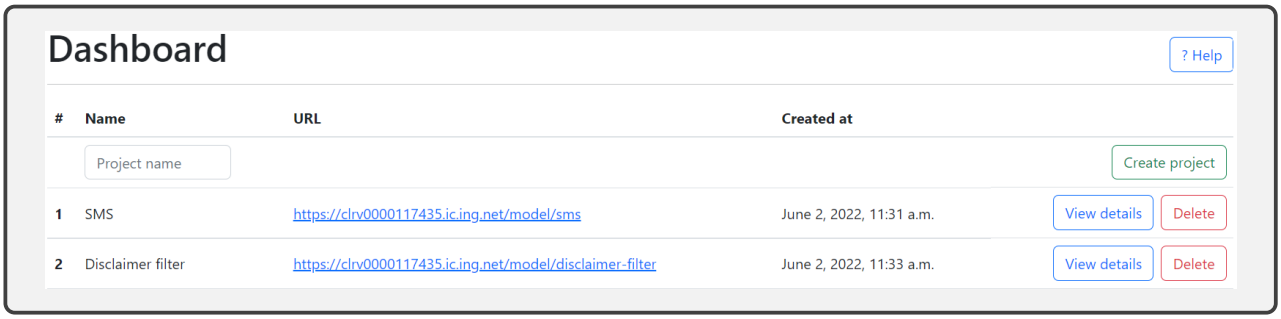

Figure 4.4: A screenshot of the dashboard page.

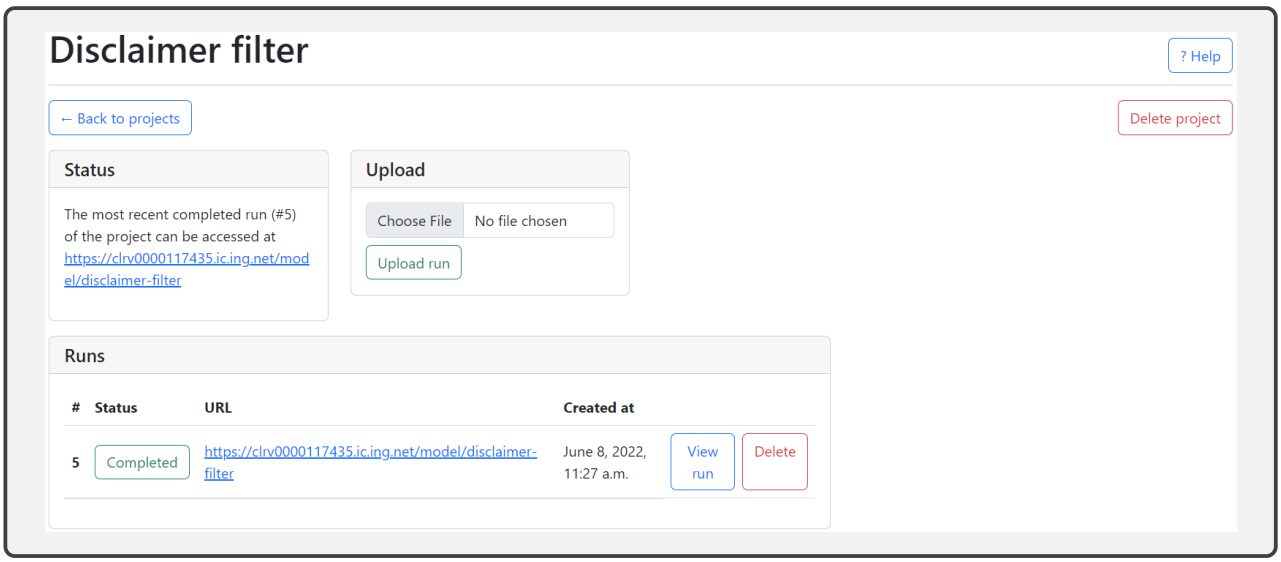

Figure 4.5: A screenshot of a project-specific page.

storage section below displays some information on the storage size of certain aspects of the run and allows the user to delete those individual aspects. The *Run logs* section on the right shows all the logs from the executed commands as well as application logs such as building the Docker images. Section 4.4.2 further describes the process of uploading and executing a model.

4.4.2. Back-end

The back-end performs the heavy-lifting of the application. Executing the ML lifecycle commands will often require significant time and resources, and are therefore performed in the background. The Python package Celery[6] is used to simplify running background tasks. By making use of Docker[7], we can run projects in an isolated and reproducible manner. The following sections will go further into detail about some of the design choices.

Python Environment

One of the most significant limitations is the prohibition of internet access. There are several options for mitigating this limitation:

1. **Predefined dependencies** Manually install a set of dependencies and make these available to the projects.
2. **Docker** Ask users to prepare their own Docker image which has the right environment configured already and use this image to run the projects.

[6] `https://docs.celeryq.dev/en/stable/index.html`
[7] `https://www.docker.com/`

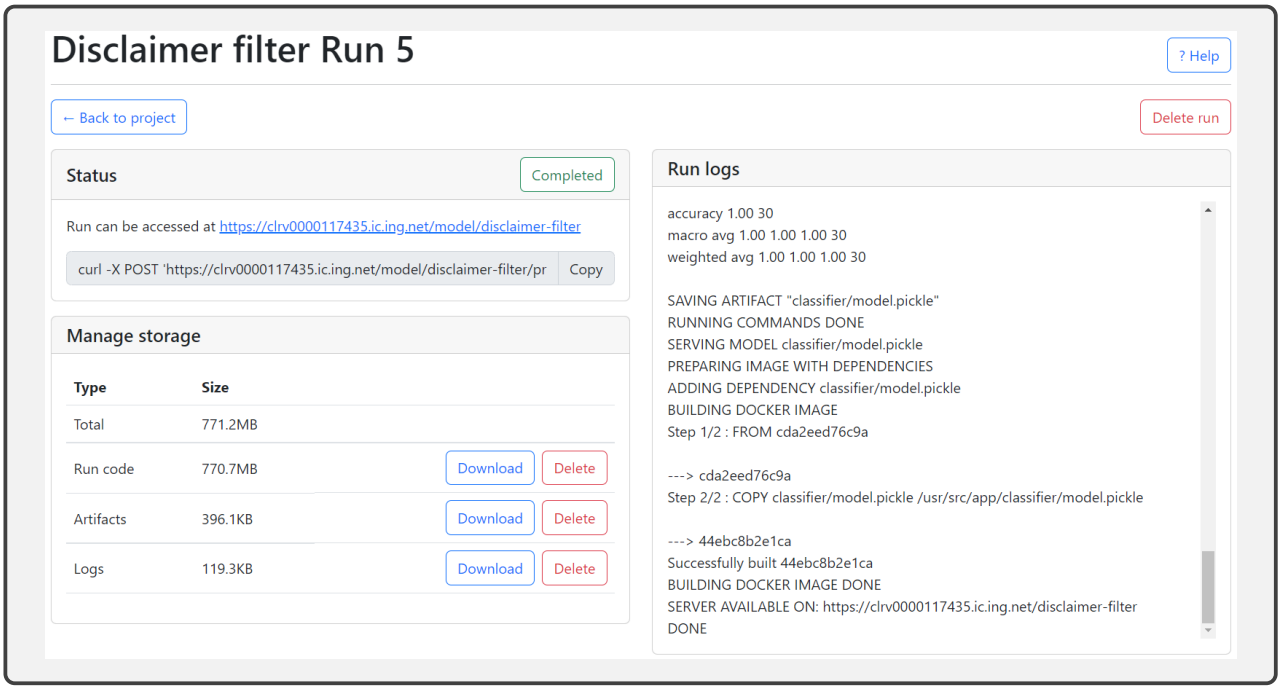

Figure 4.6: A screenshot of a run-specific page.

3. **Conda-Pack** Ask users to download and export their Python environment and send it along with the project code. A common way to do this is to use the Conda-Pack tool[8].

The first option significantly limits the generalisability of the application; users would be restricted to a limited set of dependencies and their versions. Therefore, while the second and third options require more operations-like work from the user, they are still a far better choice than the first option.

While option two provides the user more freedom to configure their environment and possibly additional required tools, it requires some level of expertise and manual work to set this up for each project, defeating the purpose of creating a deployment solution that can be operated by data scientists.

The third option still requires some manual work but unfortunately, this is unavoidable with the current set of limitations. The manual work involved in option three is still significantly less than for the second option. Therefore, the third option is preferable overall.

Project Configuration

To be able to generalise to different projects, the application needs to be flexible in terms of executing a model's lifecycle. To do so, the application requires users to create a configuration file. This file needs to be written in the YAML[9] language. YAML is a human-friendly data serialisation language commonly used for configuration files. For simplicity, the name of this file is fixed to `metis.yaml`.

Listing 4.1 depicts the template configuration file that can be downloaded from the help page. The file includes several comments to aid the user in setting up the project's configuration. First of all, users can specify a list of commands that need to be executed to train a model. This list can, for instance, consist of commands to first retrieve data and then train the model. Users can also specify environment variables that should be made available during the execution of a command. The following sections will go into detail on other configurable aspects.

[8] `https://conda.github.io/conda-pack/`
[9] `https://yaml.org/`

```yaml
# This is a template for a Metis configuration.

# Define your global environment variables here.
# These variables will be available during the execution of all commands.
environment:
  GLOBAL_ENV_VAR: global_env_value

# Use this if you want o override the filename of your packed Anaconda environment.
# The default is conda_env.tar.gz (make sure the tar is in the zip you upload).
# conda_env: conda_env.tar.gz

# Here you can define the commands that need to be executed
# before the model can be served.
commands:
  - command: python fetch_data.py
    # Use this if you want to run the command from a subdirectory in the project.
    # working_dir: some_dir
    # These environment variables will only be available
    # during the execution of this specific command.
    environment:
      COMMAND_ENV_VAR: command_env_value
    # Specify the artifacts (files/directories)
    # that should be saved after running this command.
    outs:
      - dataset
  - command: python train.py
    # Specify the artifacts of previous commands that this command depends on.
    # The artifacts will be loaded before running this command.
    deps:
      - dataset
    outs:
      - model.pickle

# Here you can configure how your model should be served.
serve:
  # You can provide a custom command that serves your model API.
  command: python serve.py
  # Or you can simply provide the path to a pickle of your model.
  # The application will call your pickled model as follows: `model.predict(data)`,
  # where `data` is a list of items that need to be predicted.
  # model_path: model.pickle
  environment:
    SERVE_ENV_VAR: serve_env_value
  deps:
    - model.pickle
```

Listing 4.1: Template configuration file.

Uploading

Once a user has packed their Python environment and configured the `metis.yaml` file, they are ready to upload everything to the application. To do so, they are required to compress the project code, the packed Python environment and the `metis.yaml` file together in a zip archive. The help page in the front-end UI provides instructions on creating the zip archive.

Artifacts
For each command, the user can specify a list of dependency and output artifacts. Output artifacts, configured with `outs`, are files or directories that need to be saved to a persistent storage after executing a command, for example, a folder containing downloaded data. Analogously, dependency artifacts, configured with `deps`, are files or directories that a command requires to execute successfully and thus need to be retrieved from a persistent storage. In the current setup, the persistent storage consists of a simple file storage that is directly mounted on the development server.

Isolated Command Executions
Each command that the user defines in the configuration file is run in a separate Docker container. This provides several benefits. First of all, it isolates each command execution from other commands, both from commands of the same model and from commands of different models running concurrently. This way, a single failing command will not have any impact on other commands. Secondly, it lays the groundwork for possible future work in terms of distributing the workload of certain commands.

Model Serving
Users can also configure how their model will be served once all commands have been executed successfully. There are two options available. The first option is providing a custom command that will start an API server that serves the trained model. The second option is providing a path to a *pickle* of the model. Pickling[10] is a process often used by ML experts to serialise and store an object. This allows them to save a trained model for later use.

When a user chooses the second option, the application takes care of starting an API that serves the model. The API consists of a single script that receives incoming HTTP requests, invokes the model and returns the model's output. This does require users to include a predefined method signature in their model so the API is able to invoke the model. The method signature is as follows: `def predict(data: list) -> list`. By making use of this option, users are not required to implement their own model serving approach which will make the life of a data scientist easier. This option also avoids duplicating code between projects to serve a model. The API currently consists of a single socket and does therefore not scale to large amounts of invocations. However, this serves as a proof of concept and calls for future work.

When users opt to let the application take care of serving the model, the application can make use of the knowledge it has on the API. For instance, the application will know the API's exact URL path that the model will be hosted on. This allows the application to present a pre-configured cURL command to the user that will invoke the model with some dummy data. This provides the user with a starting point for invoking the model through the API.

Whichever option the user chooses for serving the model, a URL will be made available for the project. This URL will directly point to the exposed port of the container that serves the model. This gives users the freedom to implement a more complex API with multiple endpoints as well. The URL structure is as follows: `https://$HOST/model/$PROJECT_SLUG`, where `$HOST` is the hostname of the application and `$PROJECT_SLUG` is the slug of the project name. To reduce manual work when deploying a new version, a project's URL will always point to the most recently completed run. This avoids users having to update any other systems that depend on the model.

Besides the latest version, older versions that have not yet been deleted will be kept running as well. These older versions are accessible by using the following URL format:
`https://$HOST/model/$PROJECT_SLUG-$RUN_ID`, where `$RUN_ID` is the unique ID of the run. This allows users to directly compare different versions. A more sophisticated use case would be to run A/B testing setups where one group would get results obtained from version A of the model and another group would get results obtained from version B. This would allow for a detailed analysis of the impact of either version. Once a run is deleted, it can no longer be accessed through the URL.

4.4.3. Deployment
The application itself can currently only be deployed manually. The manual deployment process consists of packing the Conda environment including the application's Python dependencies, zipping the application code together with the packed environment and uploading it to the development server that the application is hosted on. Once uploaded, the application code and packed environment have to

[10] https://docs.python.org/3/library/pickle.html

be extracted to the proper location and the front- and back-end services have to be restarted to make sure none of the processes still use an old version.

4.5. Evaluation

To validate the application, we perform an evaluation. This section describes the evaluation method and its results.

4.5.1. Methodology

The evaluation of the application consists of two one-on-one evaluation sessions with members of the team at ING. Both sessions took approximately one hour. The sessions took place online and were recorded to allow analysis afterwards. Both participants were also part of the interviews in the case study.

The first session was with a data scientist from the team. During this session, the participant was instructed to first get a basic understanding of the application, followed by uploading a small sample project. After succeeding, the participant was asked to consume the uploaded model. Finally, a short discussion was held regarding the value of the application for the participant and the general impression of the application.

The second session was with the chapter lead. In this session, the host described all the features of the application and performed a short demonstration of uploading and managing projects in the application. Both during and after the demonstration, a discussion was held on the value of certain features and the prospects of the application.

4.5.2. Results

Both participants made remarks on several aspects of the application. This section will go through each of these aspects and discuss the feedback provided by the participants.

Python Environment

Section 4.4.2 describes the design choice for dealing with the prohibition of internet access. Unfortunately, Conda environments that have been packed on a Windows-based OS do not transfer to a Unix-based OS[11]. Since the server that this application is hosted on is running RHEL, this approach works fine for users running a Unix-based OS (macOS, Linux, WSL, etc.) but it does not work for users running a Windows-based OS. The first participant was running Windows 10 on his local machine and was therefore not able to pack the required Conda environment. This means that he will not be able to upload projects with ease himself with the current setup of the application.

This issue was also discussed with the second participant. He proposed the idea of making sure all his team members are running the same setup, i.e. everyone running a Unix-based OS. While this would avoid the problem, it was quickly decided that this is not a feasible solution.

Azure DevOps

The evaluation meetings only included a demonstration of manually uploading a project to the application. Both participants expressed the value they see in making the connection between Azure DevOps and the application. This connection would be able to integrate the application into the CI/CD of an ML project. Once a change is pushed to a project, the project would automatically be deployed to the application. With this connection in place, they foresee several additional features that can be added to the application, such as linking project runs to their respective version in the source code.

Data

The first participant brought up the concern of how the application handles large amounts of data, i.e. several 100GB of data. While the application has access to approximately 300GB of storage, there is currently no method in place to manage this data efficiently nor is the application able to prevent a project from downloading too much data. The second participant showed less concern for a storage shortage since the storage can easily be extended, but he did see a lot of potential in optimising stored data.

[11] `https://conda.github.io/conda-pack/#caveats`

The original pipeline project that was analysed in Chapter 3 downloaded data from an on-premise instance of Relativity[12]. The models that this team at ING creates often make use of data coming from Relativity. Both participants saw great value in creating built-in support for downloading data from Relativity to simplify that process.

Other Environments
The original requirements stated that the application would first only be deployed to the development environment. Nonetheless, both participants expressed their interest in seeing the application being deployed to the acceptance and finally production environments in future work as well. The main reason for this was that the acceptance environment is a direct replica of the production environment and therefore allows for testing the application with real customer data.

Access Control
The second participant brought up the subject of access control. In the current application, there is no functionality for restricting access or limiting a user's permissions. The participant suggested that a Role-Based Access Control (RBAC) system would be a good addition to the application. With this system in place, managers would be able to decide which team member is allowed to perform specific actions within the application. Especially when this application would be deployed to the production environment, additional security measures should be implemented, such as Multi-Factor Authentication (MFA).

Model Versions
Section 4.4.2 describes how older versions of the model are still accessible after a more recent run has been completed successfully. While the second participant acknowledged the benefits of that design choice, he explained that another approach would suit their use case better. He suggested always only having one active version of a model and allowing the user to choose which version should be active. This approach would have an advantage over the current one when a user would want to roll back to a previous version. In this approach, a user would only have to switch the currently active version to an older one. Whereas in the current approach a user would have to delete the most recent run and re-upload that version manually again when they want to use that version anyway.

User Experience
Although this was not the main goal of the sessions, there were some remarks about the User Experience (UX) of the application. The first participant mentioned some minor details on the UI that could have improved the overall UX of the application and he did not find the configuration file to be very intuitive.

Final Verdict
The final verdict of both participants was positive towards the application. The first participant specifically saw the value of the application in his daily workings. He mentioned that this application would "fix the deployment challenges" he encounters during his work. While both participants saw value in the current set of features, they foresee a lot of possibilities with the application.

4.6. Discussion
The results section described the feedback obtained from the evaluation. This section discusses these results and some of the application design choices.

Python Environment
The results section describes how a Windows user is currently unable to pack a Conda environment with relative ease. Section 4.4.2 talks about two different approaches for allowing users to upload their Python environment manually. For Windows users, while still being more complex and time-consuming, the approach of requiring them to create their own Docker image would not have resulted in any impediments. Future work could include supporting both approaches. Fortunately, since CI/CD pipelines are almost always run on Linux-based servers, this limitation will not present as an issue for CI/CD pipelines.

[12] `https://www.relativity.com/`

A workaround for this issue would be to instruct Windows users to download and pack the Python environment in a local Docker container and store this packed environment on disk. A detailed explanation of this process could be added to the application's help page to simplify this process for users. This would allow all users to make use of the application. Unfortunately, this would add an additional requirement for the user of having Docker installed.

Another approach to dealing with this issue is implementing the second *Python Environment* option described in Section 4.4.2 as well. This could be implemented by letting users push custom images to ING's container registry and make the application download the images from there. By having both options implemented, users could choose whether they want the freedom of their own Docker image or the low effort of simply packing their Conda environment. Additionally, the application could support uploading and using a tar of a custom image. This would avoid users having to publish their image to ING's container registry and would thus be a quicker solution for users wanting to quickly get a project up and running.

Azure DevOps

During the evaluation, both participants talked about Azure DevOps with respect to the application. It was originally planned to connect the application to Azure DevOps. However, the application is hosted in the development environment and the connection between Azure DevOps and the development environment was broken during the development stages of this study. It was considered out of scope to repair the connection between Azure DevOps and the development environment. This is the reason why the application is currently still being deployed manually and the evaluation did not include a sample project that got deployed to the application through Azure DevOps. Once the application gets deployed to the acceptance environment or the development environment is repaired, both these issues will be resolved.

Data

The way data is currently managed in the application is not ideal for extremely large amounts of data. For instance, there is currently no way to reuse data that has already been downloaded once before. Consecutive runs will simply re-download and store the data they require. This will quickly lead to the application running out of available storage. Due to limited time, the decision was made to focus on exploring other features, rather than further optimising the management of data.

A first step in mitigating this limitation would be to implement a more sophisticated approach to managing data. An example tool that could be of great help in this task is Data Version Control (DVC)[13]. DVC is designed to handle large files and data sets. Similar to this application, DVC requires users to specify what data each step of the pipeline depends on and outputs. Additionally, it requires users to configure a storage location where data can be stored and retrieved from. When commands are executed, DVC will upload and download the required data if necessary. Since one of the main goals of this application is to make it accessible for data scientists, it would have a contradicting effect to require users to fully understand and configure DVC for their project as well. Therefore, a suitable solution would be to have the general DVC configuration contained within the application and only require users to specify the dependency and output data for each command.

Another limitation is that data that has to be retrieved from outside the ING network, has to be retrieved beforehand and uploaded along with the project code and dependencies. A workaround for this issue could be to host the data somewhere within the ING network so the application can reach and download it. However, when this is not possible, a user would have to include the data with every run upload. An improved data management system should also take this use case into account.

Most models from this team retrieve their data from Relativity. Unfortunately, it was not possible to include one of these models in the evaluation. The reason for this is that each Relativity instance is hosted in one of the DTAP environments and can thus only be accessed from servers within that environment. During the development phase of this study, the Relativity instance in the development environment was broken and could therefore not be used. Since the original pipeline project was hosted in the acceptance environment it was able to download data from Relativity. This means that once either this application gets deployed to the acceptance environment or the development Relativity instance is repaired, this application should be able to retrieve data from Relativity as well.

[13] `https://dvc.org/`

MLOps solutions need to be prepared to be integrated with different data orchestration solutions. This will be challenging since each data platform will likely require its own specific way of interacting with data. Generalising this to other organisations will be even more challenging. To try and mitigate this challenge, it is important that organisations strive to conventionalise data interaction within their organisations.

Model Versions
The discussion on model versions during the evaluation shows that model versioning in ML is important not only in terms of artifacts but also at the hosting level. MLOps solutions need to be prepared to provide hosting solutions that are able to incorporate the user's wishes on a per-version basis.

ICHP
During the design phase of the application, the use of Kubernetes[14]-like tools was also investigated. ING hosts an instance of Red Hat OpenShift[15] within IPC, called ING Container Hosting Platform (ICHP). OpenShift is a container platform that works on top of Kubernetes. ICHP would have provided a lot of interesting opportunities for the application. However, ING requires rigorous procedures to be able to use ICHP and this team did not yet complete these procedures and was thus not able to use it. Therefore, it was considered out of scope to integrate ICHP into the application.

Research Question
We can now answer **RQ3** Can we have a deployment solution for ML models in a highly-regulated context that can be operated by data scientists? By building a custom application, we have shown that we can have such a solution. This application is able to deploy ML models to an environment at ING and through evaluation, we have shown that it can be operated by data scientists.

4.7. Threats to Validity

One of the main threats to the validity of building the application is the limited evaluation. Only one of two models developed by the team could be deployed successfully to the application. The other model required data from Relativity and was therefore not able to retrieve data due to the Relativity instance not being set up properly in the development environment. A small ML project that was not developed by the team was used as additional evaluation but this has less significance than one developed by the team itself. The missing link between Azure DevOps and the application also contributes to the limited evaluation possibilities. This can also be traced back to the improperly configured development environment.

Additionally, although the thesis was intentionally scoped to this team only, we can only theorise about applying the results to other departments or companies in general. Future work should include performing similar evaluations with other teams within ING and possibly other companies as well. A member of the Global Data Integration Layer (GDIL) team at ING has already expressed their interest in this application.

[14] https://kubernetes.io/
[15] https://www.redhat.com/en/technologies/cloud-computing/openshift

5 Conclusion

MLOps faces several additional challenges compared to the traditional DevOps model. The goal of this thesis is to shed light on MLOps challenges encountered by practitioners. To do so, we performed a case study on one of the teams at ING. By conducting interviews and inspecting code manually, we managed to extract several challenges and document the approach that the team took for their ML pipeline. According to the interviews, some of the challenges were organisational, ranging from prioritising innovation to educating team members on MLOps practices. Others were more practical, such as by operating in the fintech domain, the team at ING had to deal with strict regulations, which complicates, among other things, the handling of confidential data. After this case study, we built the first version of an application that aims to mitigate those MLOps challenges. The application is able to deploy ML models to one of the servers in ING's development environment.

This research aids practitioners in their future decisions by providing an example of an ML pipeline and the challenges that came with developing this pipeline. Additionally, this thesis provides a detailed design of an ML pipeline application. While the scope of this thesis was to build an application specifically for this team, the design is not restricted to this team. Other teams, possibly even at other companies with similar restrictions, should be able to employ this application in their environment as well. The design of the application can aid practitioners in general in building similar applications for different use cases.

5.1. Future Work

This thesis was intentionally scoped to only include a subset of the features that were collected in the initial design phase of the application. This section will suggest directions for future work.

To get an even more refined understanding of the challenges of MLOps, future work should include more MLOps case studies. Even within ING, there are likely more teams that are dealing with similar problems. Analysing and documenting what specific challenges they run into and specifically how they handle them will provide both researchers and practitioners with more insights on MLOps challenges in the industry. These additional challenges and solutions could in turn also be applied to improve this application.

As mentioned in Section 4.6, there are several aspects of the application that can be extended. First of all, the current setup for letting the application take care of serving a model provides several possibilities. While the API currently serves models using a primitive approach, the API can be extended to be more scalable by, for instance, distributing workloads. This would make the API more suitable for production use. By maintaining that code centrally in the application, any improvements on the API would directly improve all projects that make use of this option.

Improving the API is not the only way to make the served models more production-ready. By migrating from the Docker instance that is being run on the server itself to ICHP, several adjustments could be made to improve different aspects of the application. Most importantly, ICHP would allow the application to run and manage several containers, that run a single model API in a model, in a more structured manner. This would require some form of load balancing between the different containers that serve a model but, fortunately, RedHat OpenShift has built-in support for load balancing. Addition-

ally, ICHP could be utilised to distribute intermediate commands. However, this would likely make the configuration of the application more complex and might therefore surpass the team's requirements.

During the evaluation of the application, it turned out that the application is currently not optimised for large amounts of data. Future work should include taking steps to improve the data management capabilities of the application, such as utilising the tool DVC.

Besides storing and managing data, future work could also include implementing common data retrieval methods. Most models from this team at ING retrieve data from the Relativity API. By implementing this step within the application and allowing users to configure what data they need, we can avoid duplicate code between the projects. Similar to implementing a model API within the application, several opportunities arise for extending this functionality in a general place. For instance, since models often require large amounts of data, it makes sense to distribute the downloading and processing of this data.

By making the application operable by data scientists, we reduce the effort it takes to operationalise an ML model, which was one of the challenges found in the case study. When common data retrieval methods are implemented, the application could incorporate data quality tests, which were lacking in the case study pipeline. Additionally, future work could look into metric reporting and monitoring to implement some form of quality assurance, which was not present in the original pipeline either.

Currently, the application only allows for invoking models through an API. This could be extended by other types of invocations, such as on a scheduled basis. Tools such as Apache Airflow[1] could be of assistance here.

[1] `https://airflow.apache.org/`

References

[1] Ram Ananth, Seph Mard, and Peter Simon. *Opening the "Black Box": The Path to Deployment of AI Models in Banking*. en-US. Apr. 2018. (Visited on 06/23/2022).

[2] Anders Arpteg et al. "Software Engineering Challenges of Deep Learning". In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (Aug. 2018). arXiv: 1810.12034, pp. 50–59. DOI: 10.1109/SEAA.2018.00018. (Visited on 01/31/2022).

[3] Lucas Baier, Fabian Jöhren, and Stefan Seebacher. "Challenges in the Deployment and Operation of Machine Learning in Practice". In: *27th European Conference on Information Systems - Information Systems for a Sharing Society, ECIS 2019, Stockholm and Uppsala, Sweden, June 8-14, 2019*. Ed. by Jan vom Brocke, Shirley Gregor, and Oliver Müller 0001. 2019.

[4] Denis Baylor et al. "TFX: A TensorFlow-Based Production-Scale Machine Learning Platform". en. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax NS Canada: ACM, Aug. 2017, pp. 1387–1395. ISBN: 9781450348874. DOI: 10.1145/3097983.3098021. (Visited on 06/01/2022).

[5] Eric Breck et al. "What's your ML test score? A rubric for ML production systems". In: 2016.

[6] Hsinchun Chen, Roger H. L. Chiang, and Veda C. Storey. "Business Intelligence and Analytics: From Big Data to Big Impact". In: *MIS Quarterly* 36.4 (2012), pp. 1165–1188. ISSN: 0276-7783. DOI: 10.2307/41703503. URL: https://www.jstor.org/stable/41703503 (visited on 06/10/2022).

[7] Min Chen, Shiwen Mao, and Yunhao Liu. "Big Data: A Survey". In: *Mobile Networks and Applications* 19.2 (Apr. 2014), pp. 171–209. ISSN: 1383-469X. DOI: 10.1007/s11036-013-0489-0. (Visited on 01/10/2022).

[8] Thomas H. Davenport. "Competing on Analytics". In: *Harvard Business Review* (Jan. 2006). ISSN: 0017-8012. (Visited on 06/10/2022).

[9] Veit Dinges and V. Martinez. *THE FUTURE OF SERVITIZATION : Technologies that will make a difference*. en. 2015. (Visited on 06/10/2022).

[10] Zhiqiang Ge et al. "Data Mining and Analytics in the Process Industry: The Role of Machine Learning". In: *IEEE Access* 5 (2017), pp. 20590–20616. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2756872.

[11] Elizabeth J. Halcomb and Patricia M. Davidson. "Is verbatim transcription of interview data always necessary?" en. In: *Applied Nursing Research* 19.1 (Feb. 2006), pp. 38–42. ISSN: 0897-1897. DOI: 10.1016/j.apnr.2005.06.001. (Visited on 01/17/2022).

[12] ING.com. *ING at a glance*. 2022. URL: https://www.ing.com/About-us/Profile/ING-at-a-glance.htm (visited on 01/11/2022).

[13] Meenu Mary John, Helena Holmström Olsson, and Jan Bosch. "Towards MLOps: A Framework and Maturity Model". In: *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Sept. 2021, pp. 1–8. DOI: 10.1109/SEAA53835.2021.00050.

[14] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, perspectives, and prospects". en. In: *Science* 349.6245 (July 2015), pp. 255–260. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aaa8415. (Visited on 06/10/2022).

[15] *Kubeflow*. en. URL: https://www.kubeflow.org/ (visited on 06/02/2022).

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". en. In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. (Visited on 01/10/2022).

[17] *MLflow - A platform for the machine learning lifecycle*. en. URL: https://mlflow.org/ (visited on 06/02/2022).

[18] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. “Challenges in Deploying Machine Learning: a Survey of Case Studies”. en. In: *ACM Computing Surveys* (Apr. 2022), p. 3533378. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3533378. (Visited on 06/23/2022).

[19] Kai Petersen, Claes Wohlin, and Dejan Baca. “The Waterfall Model in Large-Scale Development”. en. In: *Product-Focused Software Process Improvement*. Ed. by Frank Bomarius et al. Berlin, Heidelberg: Springer, 2009, pp. 386–400. ISBN: 9783642021527. DOI: 10.1007/978-3-642-02152-7_29.

[20] Philipp Ruf et al. “Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools”. en. In: *Applied Sciences* 11.19 (Sept. 2021), p. 8861. ISSN: 2076-3417. DOI: 10.3390/app11198861. (Visited on 02/02/2022).

[21] Ronny Schüritz and Gerhard Satzger. “Patterns of Data-Infused Business Model Innovation”. In: *2016 IEEE 18th Conference on Business Informatics (CBI)*. Vol. 01. ISSN: 2378-1971. Aug. 2016, pp. 133–142. DOI: 10.1109/CBI.2016.23.

[22] Damian A. Tamburri. “Sustainable MLOps: Trends and Challenges”. In: *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. Sept. 2020, pp. 17–23. DOI: 10.1109/SYNASC51798.2020.00015.

[23] Robert K. Yin. *Case study research and applications: design and methods*. Sixth edition. Los Angeles: SAGE, 2018. ISBN: 9781506336169.

[24] *ZenML*. URL: https://zenml.io/home (visited on 06/03/2022).

[25] Yue Zhou, Yue Yu, and Bo Ding. “Towards MLOps: A Case Study of ML Pipeline Platform”. In: *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*. Oct. 2020, pp. 494–500. DOI: 10.1109/ICAICE51518.2020.00102.