Author: Leon Radeck

Automated deployment of machine learning applications to the cloud

Master Thesis

Heidelberg University

Supervisors: Prof. Dr. Barbara Paech
Prof. Dr. Ullrich Köthe
Dr. Felix Roth

Software Engineering Group
Applied Computer Science

26.10.2020

Declaration of Authorship

I, Leon Radeck, declare that this thesis titled, "Automated deployment of machine learning applications to the cloud" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Zusammenfassung

Der Einsatz von Maschinellem Lernen (ML) als Schlüsseltechnologie in der Künstlichen Intelligenz (KI) gewinnt in der zunehmenden Digitalisierung von Geschäftsprozessen kontinuierlich an Bedeutung. Der Großteil des Entwicklungsaufwands von ML Anwendungen fällt dabei jedoch nicht auf die Programmierung des ML Modells, sondern auf die Erstellung der Server-Struktur, die für einen hochverfügbaren und fehlerfreien Produktivbetrieb der ML Anwendung verantwortlich ist. Die Generierung einer solchen Server-Struktur durch die EntwicklerInnen ist dabei zeitaufwändig und kompliziert, da umfangreiche Konfigurationen vorgenommen werden müssen. Neben der Erstellung der Server-Struktur ist es außerdem sinnvoll, neue ML Anwendungsversionen nicht direkt produktiv zu setzen, sondern zur Qualitätssicherung das Verhalten der ML-Anwendung in Bezug auf unbekannte Daten zu beobachten. Dabei soll beispielsweise die Fehlerrate, sowie der CPU und RAM Verbrauch überprüft werden. Das Ziel dieser Arbeit ist es daher, einerseits Anforderungen an eine geeignete Server-Struktur zu erheben und andererseits an einen Automatisierungsmechanismus, der diese Server-Struktur generiert, darauf die ML-Anwendung produktiv setzt und eine Beobachtung des Verhaltens einer neuen ML Anwendungsversion anhand von Echtzeit-Nutzerdaten ermöglicht. Zu diesem Zweck wird als Grundlage zuerst eine systematische Literaturrecherche durchgeführt, die untersucht, wie das Verhalten von ML Anwendungen vor ihrer Produktivsetzung unter dem Einfluss von Echtzeit-Nutzerdaten analysiert werden kann. Anschließend wird im Rahmen der Anforderungsanalyse eine Ist-Soll-Analyse in der Abteilung einer Unternehmensberatungsfirma im Automobilsektor durchgeführt und zusammen mit den Ergebnissen der Literaturrecherche eine Liste von User Stories für das Automatisierungswerkzeug ermittelt und priorisiert. Die Umsetzung des Automatisierungswerkzeugs erfolgt in Form einer Python Konsolenanwendung, die die gewünschte Funktionalität mittels Einsatz von IaaC (Infrastructure as a Code) und der AWS (Amazon Web Services) SDK in der Cloud ermöglicht. Das Automatisierungswerkzeug wird abschließend in der Abteilung evaluiert. Dabei führen die zehn Teilnehmer selbstständig vorgegebene Nutzungsszenarien durch und bewerten das Werkzeug anschließend über einen Fragebogen, der auf Basis des TAM-Modells entwickelt wird. Die Ergebnisse der Evaluierung sind überwiegend positiv und das konstruktive Feedback der Teilnehmer beinhaltet zahlreiche interessante Anmerkungen über mögliche Änderungen und Erweiterungen des Automatisierungswerkzeugs.

Abstract

The use of machine learning (ML) as a key technology in artificial intelligence (AI) is becoming more and more important in the increasing digitalization of business processes. However, the majority of the development effort of ML applications is not related to the programming of the ML model, but to the creation of the server structure, which is responsible for a highly available and error-free productive operation of the ML application. The creation of such a server structure by the developers is time-consuming and complicated, because extensive configurations have to be made. Besides the creation of the server structure, it is also useful not to put new ML application versions directly into production, but to observe the behavior of the ML application with respect to unknown data for quality assurance. For example, the error rate as well as the CPU and RAM consumption should be checked. The goal of this thesis is to collect requirements for a suitable server structure and an automation mechanism that generates this server structure, deploys the ML application and allows to observe the behavior of a new ML application version based on real-time user data. For this purpose, a systematic literature review is conducted to investigate how the behavior of ML applications can be analyzed under the influence of real-time user data before their productive operation. Subsequently, in the context of the requirements analysis, a target-performance analysis is carried out in the department of a management consulting company in the automotive sector. Together with the results of the literature research, a list of user stories for the automation tool is determined and prioritized. The automation tool is implemented in the form of a Python console application that enables the desired functionality by using IaC (Infrastructure as code) and the AWS (Amazon Web Services) SDK in the cloud. The automation tool is finally evaluated in the department. The ten participants independently carry out predefined usage scenarios and then evaluate the tool using a questionnaire developed on the basis of the TAM model. The results of the evaluation are predominantly positive and the constructive feedback of the participants includes numerous interesting comments on possible adaptions and extensions of the automation tool.

Acknowledgement

This master thesis would not have been possible without the active support of several people.

First of all, I would like to thank *Prof. Dr. Barbara Paech*, who supported me during the writing of this thesis from the very beginning with helpful advice, suggestions and constructive criticism. The conversations with you were always pleasant and positive.

I would like to acknowledge *Dr. Felix Roth*, who actively supported me within the MHP department with all my problems and gave me many helpful suggestions for this thesis.

I am also grateful for *Fabian Wittke* and his commitment in all matters concerning the work. I would also like to thank all my colleagues at MHP for their support and good cooperation.

Furthermore, I would also like to thank *Dr. Eckhart von Hahn* for referring me to MHP and for the many enjoyable talks.

I am thankful for *Prof. Dr. Ullrich Köthe* who is willing to be second supervisor.

I would like to give a thank to *Anja Kleebaum* for her support during the decision knowledge documentation.

I would also like to thank *Marcus Seiler* for his assistance in organisational matters.

I am thankful for the support of *Anke Sopka* regarding all of my problems during studies.

Finally, I am grateful for my family and my friends who have always supported me.

Contents

1. Introduction

1.1. Motivation and challenges

In the course of the industry 4.0, machine learning (ML) is getting more and more adopted to improve the efficiency of production processes and the analysis of the resulting data [23]. The application scenarios in this field are diverse. Predictive maintenance allows to plan maintenance work more efficiently by forecasting devices failures and malfunctions [22]. Autonomous driving enables automatically operating a vehicle without any interventions of a driver by analyzing the environment and predicting adequate control reactions [22]. Continuous quality assurance makes it possible to detect manufacturing issues close to their source by real-time analysis of production data [22]. The basic principle of machine learning in all of these areas is to create a model out of example data by using learning algorithms [23]. The model, which represents the acquired knowledge representation, can then be applied to new, potentially unknown data of the same type. Besides the mentioned application scenarios, machine learning can be appropriate whenever processes are too complicated to describe analytically, but enough sample data is existent [23].

To make organizations aware of the advantages of artificial intelligence and to investigate how the advances impact their businesses, the artificial intelligence department of the company MHP provides management consulting from the integration of an AI strategy to the implementation of the solution [38]. The MHP Management- and IT-Consulting GmbH is one of the leading consulting companies and a subsidiary of Porsche AG. Their focus is on the automotive industry, where consulting is offered to manufacturers, suppliers, dealers and importers. This thesis was written during a cooperation between Heidelberg University and the artificial intelligence department of MHP. Within the AI department, the productive operation of ML applications is especially important to meet the requirements of the customers. In this context, the successful transfer of ML applications to their productive operation in the cloud is associated with extensive technical hurdles. The server structure, that allows to operate the ML application has to be scalable and fail-safe. This server structure will be referred to as application infrastructure in this work. The application infrastructure consists out of numerous services that have to be configured and linked. Examples are networking, storage and computing services that require specific settings to function properly and collectively. Without tool support, this setup process is time-consuming and error-prone. The automatic creation of an application infrastructure through an automation tool would therefore provide great added value for the department. Furthermore, it would make sense not to put new ML application versions into production directly, but to observe the behavior of the ML applications with regard to real-time user data beforehand. The behavior includes for example the CPU and RAM consumption of the ML application and the error rate of the ML model.

Hence, the goal of this thesis is to collect requirements for a suitable application infrastructure and to implement an automation mechanism that generates this application infrastructure, deploys the ML application and allows to observe the behavior of a new ML application version based on real-time user data. This automation mechanism is then implemented based on the requirements to counteract the mentioned problems while being used by MHP employees in their respective machine learning projects.

1.2. Goals, methodology and contributions

The goals of this thesis are shown in Table 1.1. The first goal *G1* is to collect requirements for an appropriate application infrastructure. The second goal *G2* is to develop an automation tool that generates this application infrastructure, deploys the ML application and allows to observe the behavior of a new ML application version based on real-time user data.

Goal	Description
G1	Collect requirements for an appropriate application infrastructure
G2	Develop an automation mechanism that generates the application infrastructure, deploys the ML application and allows to observe the behavior of a new ML application version based on real-time user data

Table 1.1.: Goals of this thesis

In order to achieve *G1* and *G2*, the following steps are performed.

1. A comprehensive literature search is conducted to find out how the behavior of ML applications can be investigated under the influence of real-time user data before their release.

2. A target-performance analysis is carried out in the artificial intelligence department of the company, to identify prioritized requirements for the automation tool by revealing how the current application infrastructures look like, how ML applications are currently being put into operation and to what extent the behavior of a new ML application version is already being observed.

3. The automation tool is implemented based on the identified and prioritized requirements.

4. The automation tool is evaluated by employees of the company according to the TAM model under the aspects of perceived-ease-of-use, perceived usefulness and behavioral intention. It is then discussed whether or not the change requests should be included in the functionality of the automation tool.

The contributions of this thesis are the target-performance analysis itself and the automation tool.

2. Background

This chapter describes the background knowledge that is necessary to understand the thesis. Section 2.1 gives a brief overview over the key concepts of machine learning, because this thesis mentiones machine learning applications and models. Section 2.2 describes AWS, because the automation tool is implemented with the use of AWS services. Section 2.3 explains Docker, because containerization is used to encapsulate ML applications.

2.1. Machine learning

Machine learning (ML) is the study of computer algorithms that allows computer programs to automatically improve through experience [39]. This makes it possible to generate predictions without any pre-defined rules or calculation instructions. The basic principle of machine learning is to create an *ML model* out of example data by using learning algorithms [23]. The ML model represents the machine learning artifact that encodes the decision or prediction logic [23]. The example data are also referred to as *training data*. The more training data the learning algorithm receives, the more it can improve the ML model and reduce its error rate [22]. A property of the training data, such as a column name is called *feature* [53].

There are different types of learning algorithms to generate an ML model. In any situation where the example data contains both the inputs and outputs, *supervised learning* can be performed. This makes it possible to learn classification and regression tasks where examples are assigned to their respective labels [44]. In contrast to supervised learning, *unsupervised learning* is about learning without explicit feedback. Data can be divided into different clusters or the number of dimensions can be reduced [44]. Another type of learning algorithm is *reinforcement learning*, where the algorithm learns by interacting with its environment. Rewards are received for performing correctly and punishments for performing incorrectly. In contrast to supervised learning, this feedback is not fixed from the beginning, but dependent on the actions that are taken. Therefore, reinforcement learning algorithms can solve sequential decision-making problems [44].

The quality of the ML model can be assessed based on different aspects, such as its *performance*, *robustness*, *scalability* and *explainability* [47]. The performance of an ML model signifies how reliably the model estimates the output value [47]. Since a finite number of examples describes the totality of all possible variants incompletely, each learned model is afflicted with uncertainty [23]. The model can also be either overfitted or underfitted to the example data. It is overfit, if it performs well when using the training data, but poorly when using unknown data. When overfitted, the model is too closely matched to the examples given and irrelevant differences or statistical noise could be included in its decision [22]. In case of an underfit, the model does not fit the example data well enough and thus also does not perform well on unknown data. The *robustness* of the model indicates the resiliency of the model to inconsistent inputs, for example when their distribution is shifted [47]. The *scalability* represents the ability of the model to scale to high data volume during training. It can be measured by analyzing the execution time and hardware demand dependent on the number of examples and the dimensions

of their characteristics [47]. The *explainability* of the ML model denotes how understandable the predictions of the model are [47].

After the ML model was created, it can be applied to new, potentially unknown data of the same type. When an ML model is contained inside a software application that accesses its functionality, the application is referred to as *ML application* in this work.

2.2. AWS

AWS (Amazon Web Services) is a cloud computing provider that offers a variety of web services in the context of computing, storage and networks [11]. According to the National Institute of Standards and Technology (NIST), cloud computing is a "model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [37].

Cloud computing can be classified into three main cloud service models and four cloud deployment models [42]. The three main cloud service models are:

- *Infrastructure as a Service (IaaS)* — offers fundamental resources like computing, storage and networking capabilities, using virtual servers such as Amazon EC2 or Google Compute Engine [49].
- *Platform as a service (PaaS)* — provides platforms to deploy custom applications to the cloud, such as AWS Elastic Beanstalk or Google App Engine [49].
- *Software as a service (SaaS)* — combines infrastructure and software running in the cloud, including office applications like Amazon WorkSpaces or Google Apps for Work [49].

The four cloud deployment models are:

- *Public* — In the public service model, all the systems and resources that provide the service are located at an external service provider. That service provider is responsible for the management and administration of the systems that are used to provide the service [42].
- *Private* — In a private cloud, the systems and resources that provide the service are located internal to the company or organization that uses them. That organization is responsible for the management and administration of the systems that are used to provide the service [42].
- *Community* — Community clouds are semi-public clouds that are shared between members of a select group of organizations. These organizations will generally have a common purpose or mission [42].
- *Hybrid* — A hybrid cloud model is a combination of two or more other cloud models. The clouds themselves are not mixed together, but each cloud is separate and they are all linked together [42].

AWS follows the public cloud deployment model and offers all main cloud service models (IaaS, PaaS and SaaS).

Examples for AWS services in the area of computing, storage and network are named in the following. EC2 (Elastic Compute Cloud) and ECS (Elastic Container Service) are examples for computing services. EC2 allows to execute virtual machines, and ECS facilitates to run docker images in AWS. The service S3 (Simple Storage Service) is an example for a storage service. It allows to upload files into so-called "buckets", which represent storage locations in AWS. The service VPC (Virtual Private Cloud) is a popular networking service. It allows to logically separate specific resources in isolated sections of AWS [11].

The mentioned services can be orchestrated programmatically using a language-specific SDK (software development kit) to achieve functionality of a greater use. For each service a dedicated web API (application programming interface) exists that allows to interact with it. Theses APIs are well documented, so that the meaning of their inputs, their functionality and their outputs are comprehensible. The server structure of the services can be divided into *regions* and *availability zones*. A region represents a geographic location with multiple availability zones, whereby an availability zone consist out of one or more data centers that are equipped with redundant systems for power, network and connectivity [11].

2.3. Docker

Docker is an open source project for building, shipping, and running applications [30]. The key terms are defined in the following.

- *Docker container* — A docker container is the active instance of a *docker image* [30].
- *Docker image* — A docker image is a collection of all of the files that make up a software application [41].
- *Docker registry* — The docker image can be stored in a docker registry in order to be publicly found, accessed, and used by developers [41].

In contrast to virtual machines, docker containers only store the application itself and not the operating system [31]. Applications that run within docker containers directly interact with the kernel of the host system. Many applications can run simultaneously in isolation without running redundant operating systems or having to execute boot sequences. This is an important difference, because docker only helps to use the container technology already built into the existing operating system [30].

Working with docker containers has a variety of benefits. It is helpful for packaging software that requires a lot of dependencies, because the dependencies can be installed and uninstalled in their entirety without leaving any residues. It allows to safely run legacy applications and up-to-date applications on the same server. Furthermore, horizontal scalability can be achieved through running multiple docker containers simultaneously. Another common docker use case is to deploy a container across multiple stages from development to production, to allow for a consistent, testable environment [31].

On the other hand, performance can be reduced by overlaying network processes and communication between the containers and the host system [31]. Also, saving files before stopping a docker container causes difficulties. Although persistent storage is possible using docker data volumes, the integration is associated with difficulties [31]. Furthermore, the use of docker containers is only fully exploited when a microservice architecture is utilized. Otherwise, just the packaging functionality is used [41].

3. Literature search

To find publications that are relevant to the given research question, a systematic literature search was conducted. In Section 3.1 the research question is described, the inclusion criteria are listed and the execution of the search is explained. The results of the literature search, an overview of all relevant articles and the synthesis are detailed in Section 3.2. A summary is provided in Section 3.3.

3.1. Methodology

This literature search provides an overview over the current state of research regarding the research question "How can the behavior of ML applications be investigated before release under the influence of real-time user data?". The literature search used a combination of database searches and snowballing, which refers to the use of the reference list of an article, also called "backward snowballing", and the citations of an article, also called "forward snowballing" [29]. Snowballing is used complementarily to cover important literature that is not found by the termbased searches, as proposed in [50].

Both search methods used the search sources in Table 3.2. A large part of scientific literature on relevant topics of IT can usually be found in the online libraries of the three important scientific associations IEEE, ACM and SpringerLink. The published articles of these sources are generally of good quality. In order to expand the search results, arXiv and Google were also included in the search. ArXiv is a comprehensive source for publications in computer science, whereby it also allows non peer-reviewed articles. Google was used to identify relevant blog articles from respectable authors.

Nr.	Inclusion criteria
1	Title suggests relevance to research question
2	Abstract suggests relevance to research question
3	Article is available
4	Article is written in German or English
5	Article describes how the behaviour of an ML application can be investigated before release using real-time user data

Table 3.1.: Inclusion and exclusion criteria.

Source	URL
IEEE	ieeexplore.ieee.org
ACM	dl.acm.org
SpringerLink	link.springer.com
arXiv	arXiv.org
Google	google.de

Table 3.2.: Search sources.

Table 3.1 contains five criteria to include an article in the selection of relevant literature for this work. First it was checked whether the title of an article indicates relevance to the research question. If that was the case, the abstract was checked as well. Next, the availability of the full text was examined. If the full text was available and written in German or English, then it was read and it was verified whether the article fulfills the last inclusion criterion, which states that the article describes how the behaviour of an ML application can be investigated before release using real-time user data. If all five inclusion criteria matched, the article was declared as relevant. If one of the inclusion criteria did not match, the article was declared as not relevant.

To gain a basic understanding of suitable search terms, step one of the termbased search was to find at least two relevant articles. These two articles could then be used as the basis for constructing further search terms. For this purpose, three experimental searches were conducted on IEEE on 26.03.2020. The different search queries can be seen in Table 3.3. First, the research question was split into its parts. For the terms "behavior", "release", "machine learning", "real-time user data" and "monitor" several related terms were searched. For the term "behavior", the terms "behavior", "metric", "graph" and "statistic" were chosen, because all of them can be assigned to the area of data analysis. For the term "release", the terms "release", "deployment" and "rollout" were selected, because they are often used synonymously. The term "machine learning" is represented by its abbreviation "ML" and its generic term "artificial intelligence" or "AI". The term "real-time", is covered by the terms "real-time", "live" as a synonym and "traffic" which means the user web traffic data. For the term "monitoring" the two terms "monitor*" and "observ*" were used synonymously. The asterisk in "monitor*" and "observ*" allows for different word endings, for example "observing" and "observer". The first search query was then constructed out of the mentioned terms. For this first experimental search, all of the metadata of a publication are included. As a result, it was noticed that the abstracts of many search results contained the specified terms in a different context, for example "machine learning" as a tool for solving a problem. Also, there was no relevant article found.

Search step	Search step and readable search term
1	**Metadata:** (behavior OR metric OR graph OR statistic) AND (release OR deployment OR rollout) AND ("AI" OR "ML" OR "machine learning" OR "artificial intelligence") AND (live OR "real-time" OR traffic) AND (,,monitor*‘‘ OR ,,observ*‘‘)
2	**Title:** (release OR deployment OR rollout) AND ("AI" OR "ML" OR "machine learning" OR "artificial intelligence")
3	**Title:** "AI" OR "ML" OR "Machine Learning" OR "Artificial Intelligence" AND **Full text:** canary

Table 3.3.: Experimental termbased search on IEEE performed on 26.03.2020 with no search limitations.

Consequently, a more general search with focus on the publication title was performed. At this time, the search query focused on the terms "deployment", "release" and "rollout" in connection with "ML", "machine learning", "AI" and "artificial intelligence". Now, only the title was used as a search criteria, to limit the result set to articles that contain a combination of the search terms in the title. One relevant article [40] was found. Another article was still missing to

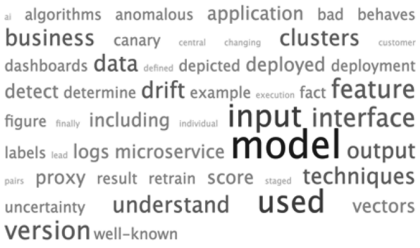
provide a basis for further search queries. The article [40] mentioned “canary deployment” as a way to observe an ML application under the influence of real-time user data. Since “canary” is an unusual term, it was used as a search term for the full text in the next search query. The terms “ML” and “AI”, as well as their full words are used for the title, to find only relevant articles in the domain of machine learning. The next relevant article [17] was found.

The first three experimental searches served to gain an overview of the literature and to get an understanding about the use of important search terms. So far, the search yielded two relevant articles. The abstracts and the relevant full text extracts of [40] and [17] were then used to generate tag clouds. Tag clouds are visual aids to display frequently used terms. The more often a term appears in the text, the larger it is displayed. Figure 3.1 shows the different tag clouds of the articles for their abstracts and full text extracts.

(a) Tag cloud of abstract of [40]

(b) Tag cloud of relevant full text extracts of [40]

(c) Tag cloud of abstract of [17]



(d) Tag cloud of relevant full text extracts of [17]

Figure 3.1.: Tag clouds of abstracts and relevant full text extracts of [40] and [17]

A new search query was then constructed based on the keywords of the titles, abstracts and full texts of [40] and [17]. The most frequent terms are selected in each case. The terms “data” and “system” were omitted, as they were very general. For the title, the search terms “ML”, “AI” and their full words “Machine Learning” and “Artificial Intelligence” were used, because to this point, both relevant articles contained “AI” or “ML” in their title. The terms “monitoring” and “production” were picked from the tag cloud of the abstract of [40] in Figure 3.1a, as they were used frequently and fit to the research question. The term “monitoring” was used, because the research question asks about behavior monitoring and the term “production” was selected, because the release of an application is often times reffered to as making it ready for production. The terms “model” and “application” were picked from the frequent terms of the abstract of [17] in Figure 3.1c, because the research questions contains the word “application” and because an ML application uses an ML model for predictions. The tag clouds of the relevant full text extracts in Figures 3.1b and 3.1d share the terms “model”, “input” and “feature” and “data”. The term “data” was again omitted, as it is very general. Finally, the terms for the title (“ML”, “Machine Learning”, “AI” and “Artificial Intelligence”), the terms for the abstract (“model”, “application”, “monitoring” and “production”) and the terms for the full text (“model”, “input” and “feature”) were used to construct the search query for search step four in Table 3.4. The terms for the title and the abstract were combined with the operator

OR, because they appeared in the two different titles and abstracts independently. The terms for the full text were combined with AND, as they appeared in both full texts.

Search step	Readable search term
4	**Title:** "AI" OR "ML" OR "Machine Learning" OR "Artificial Intelligence" **Abstract:** model OR application OR monitoring OR production **Full text:** model AND input AND feature
5	Terms for title and abstract stay the same **Full text:** model AND input AND feature AND (behavior OR behaviour) AND detect AND traffic

Table 3.4.: Termbased search on IEEE, performed on 29.03.2020, using specific keywords of the relevant results.

Because the result was still too large with 4203 results, the search term had to be adjusted. For further concretization, the relevant text passages of both full texts were examined with a text analysis tool for terms that occur in both full texts. The terms "detect", "traffic" and "behaviour" or "behavior" were used in both full texts. These terms were appended to the full text search term and search step five was performed.

The search result still contained 372 hits, thus it had to be narrowed down, to evaluate all search results. In one article, the term "traffic" appeared only in the sense of car traffic. Therefore the term was removed. The term "deploy" was used instead, as it appeared in all relevant full texts found so far. It was tried to limit the search results by restricting the publication dates and index terms of the search results, but the number of results was still 290 and thus still relatively high. That's why, all titles of the previous search results were checked for common phrases. Articles that dealt with the use of certain machine learning techniques were not relevant. Thus, the search term was further adjusted by excluding phrases like "based on machine learning" or "using machine learning". The resulting search term can be seen in Listing 3.1. The search results were then limited to articles that were published between 2010 and 2020. Some index terms were also excluded ("pattern classification", "support vector machines", "neural nets", "regression analysis", "5G mobile communication").

```
In Title: "AI" OR "ML" OR "Machine Learning" OR "Artificial Intelligence" NOT
    "using machine learning" NOT "machine learning based" NOT "based on
    machine learning" NOT "machine learning techniques" NOT "machine learning
    approach"
In abstract: model OR application OR monitoring OR production
In full text: model AND input AND feature AND (behavior OR behaviour) AND
    detect AND deploy"
```

Listing 3.1: Search term used on ACM

To expand the set of relevant articles, forward and backward snowballing was performed.

Next, a termbased search on ACM was performed on 02.02.2020. The term in Listing 3.1 was used. The publication dates for articles were limited to "2015 - 2020".

Afterwards, a termbased search was performed on SpringerLink on 04.04.2020. The input mask of SpringerLink did not allow a combined search of title, abstract and full text. Therefore, the search terms for title, abstract and full text of the term in Listing 3.1 were combined with `AND` and a full text search was conducted.

The next search took place on arXiv on 04.04.2020. ArXiv did not provide a full text search, so the search term for full text is in Listing 3.1 was omitted. The publication date was set to "2015 - 2020", the discipline was set to "computer science" and the search was performed.

Finally, a search was conducted via Google on 04.04.2020. The same search term as for SpringerLink was used.

3.2. Literature results

The results of the termbased searches can be seen in Table 3.5. The search on IEEE delivered three relevant articles [40], [17] and [53] out of 230 search results. The search on ACM did not deliver any new relevant publications out of 210 search results. One relevant book [3] and one relevant article [36] could be found on SpringerLink. The number of search results was 570, but only 50 results could be checked, as the majority of articles was not available publicly or through the university access. The search on arXiv returned two relevant publications [2] and [16] out of 424 search results. Finally, the search on Google returned one relevant web article [45] out of around 50 million results, where only the first 20 results were checked. Overall the termbased search returned 8 results. In summary, the combination of termbased search and snowballing has delivered a satisfactory amount of relevant articles. However, five search engines had to be used for the result and the review of the extensive result sets was very time consuming. Also, the individual adaptation of the search terms to the logic used by the search engines was laborious and not immediately comprehensible. An export function of the search results was only available at IEEE and ACM. It simplified the documentation of the relevance assessment with Excel considerably. The command search under IEEE had the highest flexibility among all search engines, but its operation was also associated with a high learning curve. The use of tag clouds to identify frequently used terms proved to be helpful. The construction of the resulting search terms could thus be well justified.

The results of the snowballing can be seen in Table 3.6. Article [15] was found during backward snowballing of [17]. The publication [47] was identified while forward snowballing [15]. An overview of all 10 relevant articles, their authors, publication years and sources is given in Table 3.7. Even though the publication period was limited to the years 2015 to 2020, it can be seen that 8 out of 10 articles were published in the years 2020 and 2019, which suggests that the research area is of relevance to the present situation.

Source	# Results	# Checked	# Relevant
IEEE	230	230	3
ACM	210	210	0
SpringerLink	570	50	2
arXiv	424	424	2
Google	50 million	20	1

Table 3.5.: Termbased search results

Search direction	Article	# Results	# Relevant results
Forward	Towards Enterprise-Ready AI Deployments Minimizing the Risk of Consuming AI Models in Business Applications [40]	1	0
Backward	"	5	0
Forward	The ML test score: A rubric for ML production readiness and technical debt reduction [17]	3	0
Backward	"	19	1
Backward	Machine Learning Testing: Survey, Landscapes and Horizons [53]	292	0
Forward	"	21	0
Backward	TFX: A TensorFlow-based production-scale machine learning platform [15]	21	0
Forward	"	42	1
Backward	Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology [47]	138	0

Table 3.6.: Snowballing based on relevant results, performed on 02.04.2020.

Title	Author	Year	Source
Towards Enterprise-Ready AI Deployments Minimizing the Risk of Consuming AI Models in Business Applications [40]	Muthusamy, V. Slominski, A.	2019	IEEE
The ML test score: A rubric for ML production readiness and technical debt reduction [17]	Breck, E. et Al.	2017	IEEE
Machine Learning Testing: Survey, Landscapes and Horizons [53]	Zhang, Jie M. et al.	2019	IEEE
Tfx: A tensorflow-based production-scale machine learning platform [15]	Baylor, D. et al.	2017	ACM
Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology [47]	Studer, S. et al	2020	arXiv
A taxonomy of software engineering challenges for machine learning systems: An empirical investigation [36]	Lwakatare, Lucy E. et al.	2019	SpringerLink
Practical DataOps [3]	Atwal, H.	2020	SpringerLink
Towards Automating the AI Operations Lifecycle. [2]	Arnold, M. et al.	2020	arxXiv
Engineering AI Systems: A Research Agenda [16]	Bosch, J.	2020	arXiv
Continuous Delivery for Machine Learning [45]	Sato, D. et al.	2019	Google

Table 3.7.: All relevant articles with their authors, publications years and sources.

The overview table for the relevant articles can be found in the appendix. It is split into the Tables A.1, A.2 and A.3. In the overview table, a row is created for each publication used. Each cell in the row contains information concerning the content of the publication, including abstract, author keywords, context and motivation, research questions and problems, principal ideas and results, as well as the contribution of the article. The keywords column is based on the author keywords of the publications. The publications [3], [2] and [16] did not contain author keywords, so the different chapter headings were used to extract them. The context and motivation provides background information and briefly explains what motivated the research direction. The research questions and problems column provides a basic understanding of what will be answered or solved in the examined article, as well as problems that occured during the research. The principal ideas and results column contains a description of the research process and the results that represent answers to the research questions or solutions to the mentioned problems. Finally, the contribution column describes how the article helps others in their work.

In the following, the synthesis is explained, which presents the findings of the individual relevant publications. It is split into Tables 3.8 and 3.9. The monitoring context, the monitoring objectives, the particular metrics and from the authors as useful considered practices were chosen as key points. The relevant articles were then compared and categorized on the basis of these key points. The monitoring context was selected to identify the circumstances that form the setting for the monitoring. It was also selected to check whether certain monitoring objectives are targeted more frequently in a particular monitoring context. The monitoring objectives were chosen to summarize the metrics according to intended use. This subdivision further structures the findings and simplifies the reading comprehension. The metrics were selected as a key point because they provide the basis for formal comparison and evaluation possibilities. The last two key points are the percentages of articles with a specific monitoring context that mention the respective monitoring targets. These two key points were selected to check whether there is a connection between monitoring context and objective.

First, the monitoring context is explained. Six of the ten relevant articles ([40], [17], [15], [47], [3], [2] and [45]) state that monitoring before release and with real-time user data can be done during a so-called "canary deployment", five ([53], [36], [3], [2] and [16]) propose "A/B testing", two of them ([3], [2]) propose both and [45] suggests "shadowing" as a possiblity. These terms are now briefly explained.

Canary deployment represents a gradual release of a new application version where the version is first released to a subset of users. For that, a small percentage of the users that are currently using the application are forwarded to the new application version. The new application can then be monitored with actual real-time user data. The word "canary" comes from the small birds used by miners as an early-warning mechanism of toxic gases in mines. When poisonous gases overcame the birds and they stopped singing, it was time for miners to evacuate [3].

A/B testing refers to the use of statistical hyphothesis testing to compare two groups of users that use either the new application version (treatment group) or the current one (holdout group). The goal of A/B testing is to identify the preferable version based on a predefined metric, for example the purchase rate [3].

Shadowing means that the new application version is subject to the same real-time user data as the current application version [45].

Next, the monitoring objectives are explained. The monitoring objectives summarize the field of application of the metrics thematically. Table 3.9 has been created to provide a better view on the objectives and their respective metrics. The table changes the grouping from per

Ref.	Monitoring context	Monitoring objective	Recommended practices
[40]	Canary deployment	Model inputs and outputs	Thresholding
		Model age	
		Application performance	
[17]	Canary deployment	Model inputs and outputs	Thresholding, data slicing
		Application performance	
		Model age	
		Others	
[53]	A/B testing	Business performance	-
[15]	Canary deployment	Application performance	Loose thresholding, data slicing
		Business performance	
		Model inputs and outputs	
[47]	Canary deployment	Application performance	Thresholding
		Model inputs and outputs	
		Model age	
		Business performance	
[36]	A/B testing	Application performance	-
[3]	Canary deployment, A/B testing	Business performance	Thresholding
		Application performance	
		Model inputs and outputs	
[2]	Canary deployment, A/B testing	Business performance	-
[16]	A/B testing	Business performance	-
[45]	Shadowing, canary deployment	Business performance	Data slicing
		Model inputs and outputs	
		Others	

Table 3.8.: Synthesis (1).

article to per objective and addresses the origin articles. Furthermore, the last two columns provide the percentages of articles with a specific monitoring context that mention the respective monitoring target. This metric was used to further analyze what objective is important during what context.

Business performance is proposed to be monitored by seven articles ([53], [15], [47], [3], [2], [16] and [45]). To measure it, specific key performance indicators (KPIs) are monitored. Key performance indicators are quantifiable measures used to evaluate the success of an organization or employee for performance [34]. Proposed KPIs were open rate, reading time, click-through rate [53], usage rate [47], transaction time, response time, service availability [3], sales rate, click rate, time on page [2], conversion rate [16] purchase rate [45] and install rate [15]. Four

out of five articles that propose A/B testing and four out of seven articles that propose canary deployment monitor business performance.

Application performance was proposed to be monitored by six articles ([17], [40], [15], [47], [36] and [3]). RAM and CPU usage were the most frequent metrics here, used by respectively four and three articles. Other metrics were latency, throughput, execution time, inference time and disk load. Two out of five articles that propose A/B testing and five out of seven articles that propose canary deployment monitor application performance.

Model inputs and outputs were considered to monitor by six articles ([3], [15], [40], [17], [47] and [45]). [3], [15] and [17] monitor the error rate of predictions to prevent malfunctioning. [17] also monitors the occurence of NaNs (Not a Number) or infinities in that regard. [17] and [47] monitor the inputs to verify whether they conform to a predefined data schema and to make sure the feature distribution matches that of the training data to prevent performance deviation. A difference between performance regarding training data and performance during serving is called "training/serving skew" [20]. Metrics that are monitored for this purpose are

Monitoring objective	Metrics	Articles	Percentage of articles that propose A/B testing	Percentage of articles that propose canary deployment
Business performance	Open rate, reading time and click-through rate	[53a]	4/5 = 80%	4/7 ≈ 57%
	Usage rate	[47c]		
	Transaction times, response times, service availability,	[3ca]		
	Sales rate, click rate, time on page	[2ca]		
	Conversion rate	[16a]		
	Purchase rate	[45c]		
	App install rate	[15c]		
Application Performance	RAM usage	[15c], [17c], [36a], [47c]	2/5 = 40%	5/7 ≈ 71%
	CPU usage	[3ca], [15c], [36a]		
	Latency	[17c], [36a]		
	Throughput	[17c], [36a]		
	Execution time	[40c], [47c]		
	Inference time	[47c]		
	Disk load	[3ca]		
Model inputs and outputs	Error rate of predictions	[3ca], [15c], [17c]	1/5 = 20%	6/7 ≈ 86%
	Data schema match	[17c], [47c]		
	Number of features that exhibit skew, number of examples exhibiting skew for each skewed feature, match of distributions of training features and sampled serving features, statistical bias (average of predictions in a slice of data), accuracy if label is available	[17c]		
	Incoming data: quantiles, histograms, standard deviation, top-K values of most frequent features, predicted labels	[47c]		
	Distribution of outputs and their confidence over time, Input feature clusters, anomalous inputs	[40c]		
	Inputs and outputs in general	[45c]		
	Occurence of NaNs or infinities	[17c]		
Model age	Time since deployment	[17c], [40c], [47c]	0/5 = 0%	3/7 ≈ 43%
Other	Model coefficients such as ELI5 or LIME	[45c]	0/5 = 0%	2/7 ≈ 29%
	List of announcements for each dependency	[17c]		

Table 3.9.: Synthesis (2). Articles that propose A/B testing are marked with an "a". Articles that propose canary deployment are marked with a "c". Articles that propose both are marked with "ca".

the number of features that exhibit skew, the number of examples exhibiting skew for each skewed feature, the match of distributions of training features and sampled serving [17] and quantiles, histograms, standard deviation and top-K values of most frequent features [47]. [17] also computes the average of predictions in a slice of data, to get information about a potential statistical bias. [40] uses clustering to identify anomalous values within the inputs and monitors the outputs and their distribution and confidence over time. [45] also monitors all inputs and outputs of the model to prevent training/serving skew, but it was not mentioned what metrics were used. Only one [3] out of five articles that propose A/B testing monitors the model inputs and outputs. Only one article among the ones that propose canary deployment does it not.

Model age is reported to be monitored by three articles ([17], [40], [47]). As a metric, they all use the time since the model was deployed initially. Three out of seven articles that propose canary deployment monitor model age. It is not monitored by any article that proposes A/B testing.

Other metrics that were mentioned, are model coefficients such as ELI5 or Lime [45] to help debugging the model and a list of announcements for each dependency to avoid incompatibilities between infrastructure and model. These are proposed by two out of seven articles that propose canary deployment. They are not monitored by any article that proposes A/B testing.

The frequent use of A/B testing to verify business objectives may be related to the fact that A/B testing has been a tool of user experience researchers for decades [52] and is a standard way to evaluate user engagement or satisfaction [51]. Canary deployment contrarily is a means to check general quality aspects of new software versions [24], which could be a reason that it covers the monitoring objectives more evenly.

In summary, the articles that propose A/B testing mainly suggest metrics in the area of business performance and rarely consider the machine learning associated metrics in the area of model inputs and outputs. They do not monitor model age or other metrics. The articles that propose canary deployment mainly focus on the model inputs and outputs, as well as the application and business performance, but they also monitor model age and other metrics. It can therefore be stated that the articles that propose canary deployment have a higher coverage of the monitoring objectives than those that propose A/B testing and that A/B testing is almost always associated with business performance.

Lastly, the recommended practices in Table 3.8 are explained. These methods have been frequently encountered and suggested when reading the articles.

Thresholding refers to the practice of setting limits for a specific metric. It can be used to allow only high confidence predictions of a new model [40]. The thresholds can start high and then be lowered continuously after as the effects of the new model have been observed. Thresholds can be set to initiate alert notifications that report the exceeding of the specified limit to the developer [17]. [15] suggests to use loose thresholds to avoid false negatives.

Data slicing means slicing a data set along certain dimensions of interest to allow for a more fine-grained understanding of model quality. Slices should distinguish subsets of the data that might behave qualitatively differently, for example, users by country or movies by genre [17].

3.3. Summary

The aim of the literature search was to answer the research question “How can the behavior of ML applications be investigated before release under the inuence of real-time user data?”. A combination of termbased searches and snowballing was carried out and five different search sources were used. To facilitate the identification of key terms during the search, the use of tag clouds as means of visualization has proven to be useful. As a result, ten relevant articles were found, which were then summarized in an overview table contentwise and furthermore compared in a synthesis based on different selected key points. It turns out that the behaviour of an application in machine learning can be assessed against various monitoring objectives, each of which requires the examination of specific metrics. There are metrics that monitor the achievement of economic goals, metrics that focus on application performance and metrics that examine specific aspects of machine learning. In the literature, methods such as canary deployment, A/B testing or shadowing are used to enable monitoring under the influence of real-time user data. For this purpose, the user data is either split or replicated to different application versions. It was recognized that A/B testing mainly focuses on metrics that are relevant for business performance and that canary deployment has a broadercoverage of the monitoring objectives . Furthermore, many relevant articles mentioned practices to facilitate and support monitoring, such as setting thresholds for metrics and slicing the datasets along certain dimensions to improve understanding of model quality. The methodology of the literature search and its results, consisting out of the essential contents of the relevant articles and their comparison, summarize this chapter.

4. Requirements analysis

This chapter describes how a target-performance analysis in the department "Artificial Intelligence" of an IT consulting organization was prepared, how it was executed and what results were obtained. It then describes how prioritized requirements for an ML infrastructure and an automation tool were derived from the results of the analysis.

4.1. Preparation and execution of the target-performance analysis

For the preparation of the target-performance analysis, an interview guideline with six sections was created. This guideline was then followed in all interviews conducted. In the first section the interviewee was welcomed and thanked for their willingness to participate. Then the motivation for carrying out the target-performance analysis within the framework of this master thesis was explained. The interviewee was told that their name will only appear pseudonymously in connection with the answers and it was asked whether the interview could be recorded. In the second section, key terms and their definitions were explained in order to have clarity about their meaning during the interview and to avoid misunderstandings. The terms and their definitions can be seen in Table 4.1.

Term	Definition
ML model	Artifact of machine learning that is generated using training data and can make predictions about input values.
ML application	Software program that accesses the functionality of the ML model.
Deployment	Installation of an ML application on a target system.
Release	Publication of a provided ML application for use by end users.
Infrastructure	Server structure on which the ML application is provided.
Live userdata	All input data of end users during the productive operation of an ML application (HTTP requests, form input, etc.)

Table 4.1.: Term definitions.

After the explanation of terms, the next section discusses an overview of the machine learning life cycle with the interviewee. The overview image is shown in Figure 4.1. Some details have been removed for a simpler introduction. The terms "release" and "infrastructure" have been added for clarity.

The overview image of the machine learning life cycle takes up the defined terms and is intended

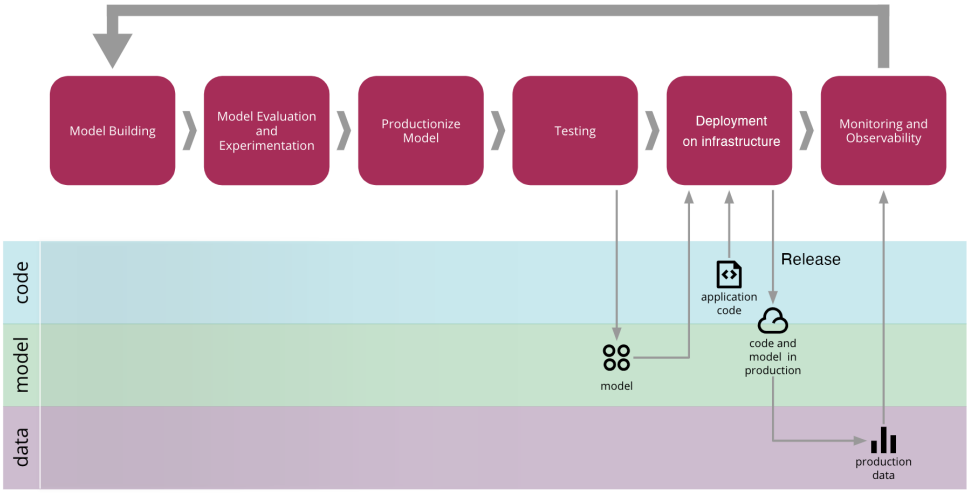


Figure 4.1.: Machine learning life cycle [45] with manual alterations.

to create a common understanding for the subsequent questions. After discussing the overview image, general questions about the person's work are asked in the next section to give an insight into the person's experience, their current tasks, problems and solutions. Additionally, these questions make it easier for the interviewee to get comfortable, since the answers are relatively easy to give. In the interview guide, general questions of this type are marked with the letter A. In contrast to type A questions, type B questions are aimed at specific topics of this work. Questions of type C refer to solutions of problems that were proposed. In the next section, the interviewee is first told that the following questions refer to their current project. The questions in this section are divided into the areas "infrastructure", "deployment", "monitoring", and "overall" and shed light on the actual state on the one hand and the target state on the other. The order of the questions correlates with the phases of the machine learning life cycle. First an infrastructure is created, the deployment can then take place and afterwards the monitoring can be applied. In the case of the infrastructure, questions about the structure, creation and non-functional requirements are asked. The questions about deployment focus on the deployment procedure and the questions about monitoring focus on the monitoring procedure, metrics and their presentation. Finally, a last question that affects all areas is asked. The list of all questions can be seen in the result section in Tables B.1, B.2 and B.3 in the appendix. In the last section of the interview guide it is asked whether the interviewee still has a question. If yes, it is discussed, if not, thanks are given for their participation and it is clarified how the interviewee will be contacted in case of subsequent questions. Finally, the farewell is said.

In order to contact potential candidates, an e-mail template was created together with the supervisor in the company and sent to a total of twelve people. Of the twelve people contacted, ten agreed to an interview. One person did not respond and the other person did not see any professional reference in the machine learning area. Meetings were arranged for the ten participants via Outlook and Microsoft Teams. An online telephone call was made with each of the ten participants, the duration of which was planned to be 45 minutes. The six sections of the interview guide (introduction, explanation of terms, view of the machine learning life cycle, general questions about the person's work, specific questions about the areas and completion) were gone through. During the interviews, key notes were made on the interviewees' answers in order to refer to the subsequent questions and to show interest. Since the interview was taped, extensive notes were not necessary. For the discussion of the machine learning life cycle, the screen was temporarily shared.

After the interview was completed, the recording was listened to again and the key points of the answers to each of the questions were documented. The names of the interviewees were written pseudonymously in form of letters next to the answers. For questions about projects, the project names were also written down pseudonymised in form of letters.

4.2. Results

The questions on project experience, role and current project are shown in Table 4.2 along with their evaluation.

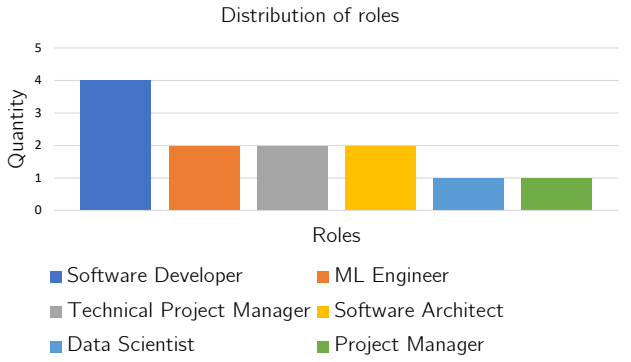
Question	Evaluation
How many ML projects have you been involved in?	 The average person interviewed has already been involved in 3 to 4 projects in the ML. The number of projects had a value range from 1 to 7.
What was your role in this project?	 A: ML Engineer B: Technical project manager C: Software developer D: Software developer E: Software developer F: Software architect G: Project manager H: Technical project manager and ML Engineer I: Data Scientist J: Software architect and developer
What project are you currently working on?	A: PV - B: PW - C: PX - D: PX - E: PZ - F: PQ - G: PS - H: PX - I: PZ - J: PY The project names were pseudonymised.

Table 4.2.: General questions about the work of the interviewee.

The average interviewee has been involved in 3 to 4 projects and the value range was from 1 to 7. Interviewee C had the most project experience with 7 projects. Interviewee F had the least experience with one project. Six different roles were mentioned. Software developer was the most frequent one. Other roles were ML Engineer, Data Scientist, Software architect, technical project manager and project manager. Each of the ten interviewees was working at one project at that moment of time. Interviewees C, D and H were all working on project PX.

The tasks, problems and solutions for each role are now described.

Software Developer

C The task was the implementation of a new use case, as well as the search and evaluation of data and its quality at the customer's site, as well as the subsequent data homogenization. Data silos or places where data cannot be found due to access restrictions, are problematic when searching for data. One problem with the homogenization of data is the heterogeneity of their formats. Data preparation is thus considered to be the most time-consuming activity. Depending on the scope of the project, setting up the CI/CD pipeline can also be extensive. The use of cloud services is seen as helpful in this context.

D The task was software development in the cloud and DevOps area. The AWS documentation was seen as problematic related to practical programming. Therefore debugging had to be used frequently.

E The task was the implementation of new features. In the context of testing, blocked ports in the infrastructure were mentioned as problematic. A test was therefore divided into several tests, which were then checked individually.

J The task was the development and integration of new features and the execution of refactorings. The communication between different systems or components was mentioned as a problem, which is often difficult to establish. The deployment procedure depends on the use case and there are various ways of its realization in the cloud. When deploying a new version of an application or model, it must be ensured that the quality does not deteriorate.

ML Engineer

A The task consisted of integrating and homogenizing data sources for later analysis while fulfilling data protection requirements with user consensus. In general, problems often arose with APIs, logging at different levels, data protection compliance during implementation, testing and data heterogeneity. Communication problems also occurred frequently.

H The task was to build the AWS infrastructure and to answer the question how generalizable the infrastructure could be. There were problems with the customer's fragmented infrastructure and the lack of knowledge about standardized procedures for accessing SAP systems. For data access, the team enquired about similar projects and contacted colleagues. In some cases, the company itself created so-called blueprints (generalizable pieces of software) that various teams in the same situation can use in the future. Furthermore, there were problems with the deployment. In AWS there is no superordinate structure to operate multi-account management. This means that it is difficult to provide many other accounts with the same software from one central account. Here, the development of individual software was helpful.

Technical Project Manager

B The task was the development of the software model as well as customer communication. One problem was the lack of clarity about the shape of the product at the end. The customer is aware that ML should be used, but it is not clear how the final product should look like. This ambiguity was tried to be solved by increased client conversations.

H The task was to define work packages and the next steps in the project (roadmap planning), to record the customer's requirements and to check the project progress. The problem is usually that the customer has no idea what AI can do and has no vision of where he wants to go. To solve this problem, workshops are held with the customer, where examples are used to explain what AI is and how it can help. In this step possible use cases and problems are discussed. A business value calculation is also carried out to clarify what value a use case has for the customer.

Software architect

F The task was to design the project architecture. There were problems with the topic data security, legal and consent scopes in relation to the DSGVO. Here, the coordination and time expenditure was very high. In some cases, teams shifted the responsibilities to each other. As a consultant, a lot of communication was required to satisfy the customer's needs on the one hand and to facilitate internal cooperation on the other.

J Conception of new features or components and integration into the given environment were the tasks.

Data Scientist

I The task was to build the infrastructure and enable deployment in the context of transferring a POC to a production system. There were problems with the versioning of services, models and data. Terraform was used to facilitate the creation of the infrastructure and the versioning of the services was done via ECR.

Project Manager

G Creating new offers for assignments and report problems was the task.

The overview in Figure 4.2 classifies the tasks of the interviewees related to the machine learning life cycle. The classification was done manually on the basis of the answers to the questions of the personal tasks. It is noticeable that the tasks focus mainly on the area of application code (B, C, D, E, F, J), infrastructure (A, F, G, H, I, J) and deployment (H, I). Cross-project tasks (H, B, G) and the handling of training data (C, A) are further areas. Tasks in the area of monitoring were not explicitly mentioned. This does not mean that monitoring has no relevance. It was simply not mentioned as a current task in the currently assigned project. According to the answers to questions 18a and 18b for example, properties of the ML model or ML application are already being monitored in four projects and nine out of ten interviewees believe that monitoring should be used in the future. This diagram can therefore only be seen as an overview of the primary tasks of the interviewees that have just been assigned and not as a general assessment of the relevance of the individual areas.

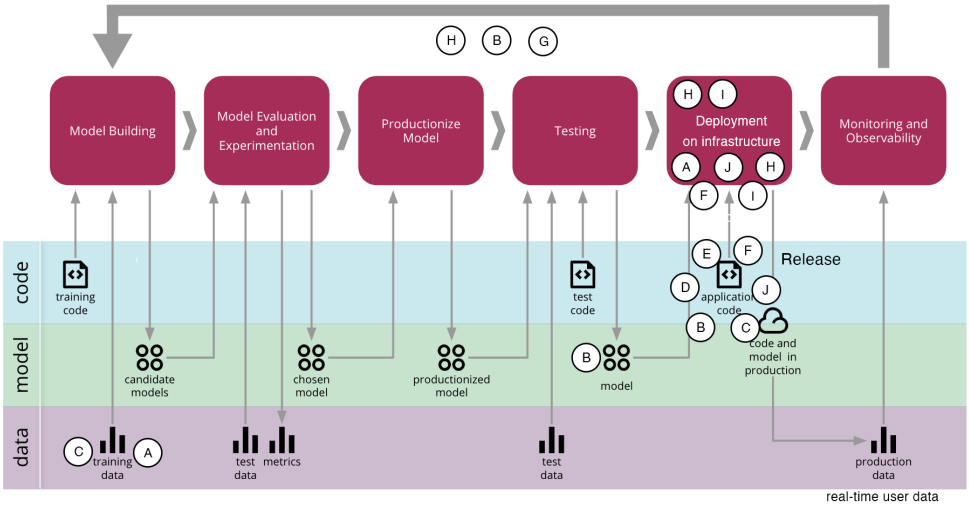


Figure 4.2.: Classification of the tasks of the interviewees in the machine learning life cycle [45].

The complete evaluations of the specific questions on infrastructure, deployment and monitoring are listed in Tables B.1, B.2 and B.3 in the appendix. These tables contain more specific details about each individual project and also link each answer to the corresponding interviewee. These details have been omitted in the following for easier reading comprehension. For examples of concrete infrastructure designs or deployment procedures, it is recommended to review the full evaluation in the appendix, questions 7a and 11a in particular. In the following, reference is made to the alphanumeric identifiers of the questions from the full evaluation in the appendix.

Infrastructure

Design of the infrastructure

Q: *What does the infrastructure for the deployment of ML applications in the current project consist of (7a) and how would it look ideally (7b)?*

A: In the interviews, it was mentioned that AWS is the primary cloud provider and that an individual infrastructure for a project can be created by orchestrating different AWS services, such as S3 as data or model storage, ELB for load balancing or ECS/ECR/EKS for deployment. Sometimes, external tools were used as well, e.g. MetaFlow for the structuring of Data Science workflows or Cloudera as a platform for data engineering. Especially for the different stages (dev, prod, pre-prod and prod) a satisfactory capacity was considered important. A wide choice of frameworks, programming languages and visualization methods was wished for as well. A quick access to notebooks during model development was also considered important. Sufficient performance regarding GPU and RAM was one optimal feature that was mentioned. AWS as a cloud provider was regarded as suitable for a quick creation of the infrastructure. One interviewee described the optimal infrastructure as a combination of an S3 bucket for data storage, Kubernetes for preprocessing and a container for training. Another interviewee said that using a pure EC2 instance would be good if less abstraction is preferred.

Creation of the infrastructure

Q: *How was the infrastructure created (8a) and would a tool be helpful to automate it (8b)?*

A: Tools such as Terraform or CloudFormation are used for the creation of the infrastructure. Of the four projects to which there was a response, three projects used Terraform and one used CloudFormation to create the infrastructure. These tools use IaC (Infrastructure as code) to define and create the infrastructure. The interviewees justified the use of such tools with a reduced workload, time savings, easier reproducibility and lower susceptibility to errors. Reference was always made to existing tools and no new form of a tool was requested.

Q: *What problems occured during the creation of the infrastructure (9a) and what solutions can be imagined (9b)?*

A: The automation of the creation of the infrastructure is not possible without appropriate know-how. The complexity of the infrastructure can be very high and the occurence of role and authorization problems is difficult to handle. Managing multiple accounts in AWS simultaneously is problematic as well. One interviewee said that there is no reasonable verification of CloudFormation templates without deployment. The necessary know-how for realising the automation of the creation could be acquired through communication with colleagues. The role and authorization problems can be handled through a trial and error procedure. The management of several accounts simultaneously in AWS can be solved by developing individual software. In order to reduce the management effort of the infrastructure, services controlled by AWS can be increasingly used.

Q: *What are the differences between the infrastructure for development of a prototype and a production application (25a)?*

A: During the development of a prototype, the load on the infrastructure is significantly lower than in a productive application. This means that the hardware costs are also lower. Furthermore, the infrastructure does not have to be connected to the Internet, as there are no external accesses. With a productive application, more emphasis is also placed on monitoring and traceability. A structured approach to version control is also more important here. Scalability and security are particularly important in a production application.

Non-functional requirements of the infrastructure

Q: *Which non-functional infrastructure requirements were specified in the current project (10a)?*

A: The most frequent non-functional requirements were scalability, mentioned by six interviewees and data protection, mentioned by four interviewees. Others were reaction time, modularity, model quality and multi tenancy.

Deployment

Procedure of the deployment

Q: *How was the ML application deployed (11a) and would a tool be helpful to automate the deployment (11b)?*

A: For the deployment of an ML application, tools like Bamboo, Concourse, Gravity and MetaFlow are used. Bamboo [14] is a continuous integration and deployment tool that ties automated builds, tests and releases together in a single workflow. Concourse [25] is most commonly used for CI/CD and is built to scale to any kind of automation pipeline. Cloud Foundry [18] is an industry-standard open source cloud application platform for developing and deploying enterprise cloud applications. Gravity [27] is an open source toolkit that provides true portability for cloud-native applications. It allows developers to package a Kubernetes cluster and all its applications into a single file called a "Cluster Image". Metaflow [26] is a human-friendly Python library that helps scientists and engineers build and manage real-life data science projects. Three projects trigger the deployment by commits. All 8 interviewees who responded think that a tool for automating the deployment is helpful. The reasons for this are time saving, automatic test execution and automatic triggering. One interviewee finds automatic deployment useful, but prefers manual deployment to have more control. Suggestions for the form of the tool were not given, reference was made to existing tools such as Bamboo, CloudFoundry and CodePipeline.

Q: *What problems occured during the deployment of the ML application (12a) and what possible solutions can be imagined (12b)?*

A: The configuration of the test and production environments, which is particularly error-prone, is mentioned as a problem during deployment. The versioning of data and model is also sometimes problematic. Five interviewees said, that there were no problems. With regard to the versioning of data, there are third-party tools. A blue-green deployment could be used to ensure model quality.

Q: *What are the differences in deployment when developing a prototype and a production application (13a)?*

A: A prototype has no contact to user and thus requires no automatic data processing. It even may not be deployed at all, when it is sufficient to determine certain evaluation metrics on the own laptop to show the customer whether a use case is doable. A prototype does also not require a release and therefore there is no release process compared to a productive application. A zero downtime deployment is also not necessary.

Quality assurance during deployment

Q: *What methods were used to check the quality of a model before going live (14a) and should these methods (still) be used in the future (14b)? Under what circumstances are these methods feasible (15b)? If no methods were used, what were the reasons for that (15a)?*

A: Offline evaluation metrics, such as precision, recall and f1 score are used to check the quality of the model during training. An alarm is triggered of certain metrics fall below

a defined limit. Functional tests are used to check if the predictions of the model are valid. Furthermore, it is checked whether the model has been trained with the correct parameters. The test results of the new and old model are compared. A dashboard with metrics on the quality of the predictions is also used. This prevents a new model from being of lower quality than a previous model. Five interviewees gave an answer and all think that quality review methods should be used in the future to detect changes in data and model quality, to maintain the quality characteristics and to guarantee correct functionality. To monitor the quality of the model, quality criteria must be defined and test data must be available. Furthermore, a reasonable versioning of the model is required, which must be compatible with the deployment pipeline. A microservice architecture is also required in many cases where traffic (real-time user data) needs to be switched between different models. Eventually, to monitor the verification of the quality of the model during production operation, there must be at least some users.

Q: *Was canary deployment, A/B testing or shadowing (16a) used? Which errors could be prevented by those methods (17a)?*

A: A/B Testing, Canary Deployment and Shadowing are not used in any current project. In previous projects A/B testing was performed. According to four interviewees, canary deployment is considered to be useful. Two interviewees have already performed a Blue-Green deployment. The lack of A/B testing or canary deployment could be related to the fact that these methods may not sufficiently known or technically difficult to implement. By using these methods, changes in the data and model quality can be detected. In addition, software problems are detected and server errors occur due to incompatibility of software and server. A/B testing can be used to assess changes in user experience. A blue-green deployment can also enable high availability.

Monitoring

Procedure of the monitoring

Q: *Are properties of the ML application or ML model currently monitored (18a)? If and why should they be monitored in the future (18b)?*

A: Four projects monitor properties of the ML application or the ML model. Nine interviewees responded that they think that properties of the ML application or model should be monitored. This will allow changes in user behavior, data distribution and model quality to be identified. If several models are used simultaneously, it is also important to monitor the query rate of the models. Scaling can also be simplified by monitoring. Furthermore, calls of the ML application for traceability and time measurement should be monitored. The error rate of the model and all important fixed KPIs should be monitored, CPU and RAM consumption are also interesting for monitoring. In general, online monitoring is more meaningful than monitoring during training. Monitoring in general is helpful for quality assurance.

Q: *How was the monitoring carried out from a technical point of view in the majority of projects (19a) and how could the procedure be made easier (19b)?*

A: In one project, the evaluation metrics of a new model are compared with the old model after each training. Thresholding is used and alarms are triggered in this context. In another project, the CPU load is measured manually in Python. In one instance, the

results of the ML application are written in tables and then displayed via frontend in the form of a dashboard. In one project an ELK stack (Elastic Search, Logstash and Kibana) is used within AWS. Elasticsearch is a distributed, open source search and analytics engine for all types of data [48]. Logstash is a free and open server-side data processing pipeline that ingests data from a multitude of sources and transforms it [35]. Kibana is a free and open user interface that lets you visualize your Elasticsearch data [32]. In one project, Splunk and Kibana are used in combination. Splunk is a scalable platform for monitoring and analyzing data [46]. The application either writes logs to files if Kibana runs on the same server. Alternatively, Kafka is used as a message broker to send log messages to the server running Kibana. The log data can then be queried via Query Language. Often call IDs, user name, component, class and message are logged. In general, all requests can be logged and displayed on a dashboard. Alternatively, certain metrics can be displayed per model. CloudWatch [19], an AWS service, can also be used for logging and alerting. Together with integration via a slack channel, the alert messages reach the recipients. The use of cloud services could facilitate monitoring, an example is the load monitoring in AWS. The use of thresholding and alerting during monitoring is also seen as useful.

Q: *What problems occured during monitoring in the majority of projects (20a)? What are possible solutions (20b)?*

A: A universal evaluation metric such as accuracy is not always useful or applicable. Additionally, in monitoring, there are access problems and the customer's specifications limit the choice of tools. The granularity of monitoring is important to avoid fingerpointing. If a service has a poor performance, it could be because it internally accesses another service, which is solely responsible for the performance. So the granularity of monitoring must be fine enough. This can be achieved by logging the complete call stack.

Monitoring metrics

Q: *What characteristics of the ML application and the ML model were monitored (21a) and what other properties should be monitored ideally and why (21b)? What errors should be prevented by monitoring (22b)?*

A: Several characteristics were mentioned. The precision, recall, f1 score, learning rate, accuracy of the model was monitored during training. The response time and quality of predictions of the model were found helpful to monitor during operation. It would make sense to monitor user behavior, data distribution, model quality and all KPIs. Precision, recall, f1 score are useful for classification problems, accuracy for numerical problems. The maximum and average of the data is useful to monitor. The frequency of the model calls and the distribution of the output values is also useful. For scalability, monitoring the CPU load is helpful. Through monitoring, the user satisfaction should be ensured and the probability distribution of expenditure should be kept satisfactory. Unrealistic inputs should be avoided and the timeliness of the data should be guaranteed. In general, monitoring is helpful for error analysis, as errors can then be better reproduced.

Metric presentation

Q: *Where were the results of the metrics presented (23a) and how should they be presented ideally (23b)?*

A: A dashboard was used in all projects that applied monitoring. It was considered to be a

satisfactory option. It should be clarified with stakeholders which metrics are presented. A clean definition of metrics is important.

Infrastructure, Deployment and Monitoring

Q: *Is there a core functionality among all projects that can be automated for infrastructure, deployment and monitoring (24b)?*

A: One interviewee said that the automation of data integration via pipelines for reproducibility would be helpful. The feedback loop from monitoring to model building is important. In general, deployment, infrastructure and testing can be well simplified by automation. Another interviewee said that an S3 bucket in combination with ECS would be a good basis for an infrastructure. One opinion was that the different test stages should be prebuilt and have guaranteed resources. It should be obvious why a build takes a long time and how long it takes to start. One interviewee said that a knowledge base for example architectures is useful and is currently being created. Two interviewees said it would not be possible. One reasoned it with the customers having strict specification. One interviewee said that data access, model development, deployment and production can be automated. Another one said that monitoring for AWS could possibly be generalized.

4.3. Discussion

The target-performance analysis provided insights into the experiences, projects, roles and tasks of ten interviewees from a department for artificial intelligence on the one hand and into their knowledge of the specific areas of machine learning infrastructure, deployment and monitoring on the other.

In the first part of the analysis, it could be determined that the average person interviewed was a software developer with the experience of about three to four machine learning projects. In addition to the software developer, five other roles were identified. Apart from the four software developers, two ML Engineers, two Technical Project Managers, two Software Architects, one Data Scientist and one Project Manager were interviewed. The identified tasks, problems and solutions of the interviewees gave a short impression of their daily work and are helpful to get a better understanding of the responsibilities of the different roles. It was noticeable that the tasks of a role were sometimes not self-evident. A data scientist, for example, had the task of creating the infrastructure, although it was assumed that his role would be more closely related to data processing. The mapping of the tasks in the machine learning lifecycle showed that at the time of the interview, the application code, infrastructure and deployment were the most important fields of activity. At this point it was suspected that the phase of model building would be of more relevance. However, the questioning of the tasks only represented a snapshot in time and therefore cannot give any statement about the general importance of the areas of activity.

In the second part of the target-performance analysis, the areas of infrastructure, deployment and monitoring were examined. Regarding the infrastructure, questions were asked about its design, creation and non-functional requirements. It was revealed that a machine learning infrastructure can consist out of different AWS services and that it can be automatically created by using infrastructure as code (IaC) tools, such as CloudFormation or Terraform. The range of services was suprisingly wide, whereby the most frequently mentioned AWS services were S3, Sagemaker, ELB, ECS, ECR and EKS, and Route53. This shows that the adoption of AWS as

a cloud provider is essential for the design of the infrastructure and that the department has a particular technical affinity in this area. Various non-functional requirements of the infrastructure were mentioned, but the most commonly called ones were scalability and data protection. This is not surprising, as these requirements are commonplace in the industry. Regarding the deployment, questions were asked about the procedure and quality assurance. The deployment process was heavily reliant on the use of external tools. For example, Bamboo, Concourse or MetaFlow were integrated into pipelines, which were partly triggered by new commits. The deployment methods identified in the literature search, such as canary deployment, A/B testing or shadowing were interestingly not used in the current projects. The lack of such methods could be related to the fact that these methods may not be sufficiently known or technically difficult to implement. However, blue-green deployments have already been carried out in some projects as a means of quality assurance. The area of monitoring was the last to be examined. In this context, questions were asked about the procedure, metrics and their presentation. The approach of monitoring differs between projects. Often external tools, such as Kibana, are used to aggregate, query and display logs. In doing so, the accuracy of logging is essential, because it must be possible to determine exactly how long which component takes to execute an operation in order to identify performance bottlenecks. Within the projects, evaluation metrics are currently being monitored especially in the training phase of the model. Examples for those metrics are precision, recall, f1 score and accuracy. During productive operation, the response time and the quality of the model's predictions are also checked. However, this is only a small part of the metrics that were identified in the literature search. The visualization of metrics on a dashboard is generally considered useful, whereby the selection of metrics must always be tailored to the individual application or model. For example, the accuracy is meaningful for numerical problems and the precision, recall and f1 score are more appropriate for classification problems. In general, determining the metrics in coordination with the stakeholders is recommended. The setting of threshold values for certain metrics in combination with alarm messages is considered helpful, which also corresponds to the findings of the literature research.

The results of the target-performance analysis provided valuable insights into the work of the interviewees. It was shown how industrial projects in the field of machine learning are realised and how modern cloud technologies play a key role in this context. The detailed answers of the interviewees could especially help non-experts with the implementation of well built machine learning projects. For specialists in the field on the other hand, the findings may provide new incentives due to the variety of different technologies involved. Overall, the conduction of the analysis is considered to be successful, because the conversations with the interviewees not only produced relevant results but also strengthened interpersonal relationships and contributed to the exchange of knowledge in the department.

4.4. Lessons learned

The execution of the target-performance analysis was carried out without any major problems. One reason for this was the thoughtful and considerate development of the interview guide together with the supervisor, whereat the exact formulations of the questions and the use of specific terms were checked multiple times for comprehensibility. The clarification of the term definitions in the beginning of the interviews prevented misunderstandings and the inclusion of the machine learning overview image ensured a common understanding between both parties. Starting the interview with general questions about the interviewee had a loosening effect on the dialogue atmosphere and is therefore considered recommendable. In some situations it was difficult to adhere to the questionnaire exactly. When solutions for problems were asked and the interviewee already mentioned a lot of problems before, it had to be decided whether a solution would be asked for each of the problems or, out of time reasons, only for a subset

of them. Sometimes it was also difficult to not let the conversation topic slip into areas that were not covered by the questionnaire. In these cases it has proven to be useful to let the interviewee finish speaking, then take up an aspect of his answer and subsequently ask the next relevant question. For the documentation of the answers to the questions of each interviewee, a text file was created each time. In retrospect, using a central document for the answers of all interviewees in the first place, would have saved a lot of time that was spent on gathering the answers of each question into one place. Regarding the evaluation, the classification of questions into sections and subsections was important to structure the questionnaire, resulting in an easier reading comprehension.

4.5. Threats to validity

The validity of the target-performance analysis denotes the trustworthiness of the results, and to what extent the results are not biased by the researchers' subjective point of view [43]. In the following, the construct validity, internal validity and external validity is discussed. Construct validity reflects to what extent the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions. The construct validity of the target-performance analysis is reinforced through the communication of the definitions of the fundamental terms in the beginning of the interview. Nevertheless, it cannot be said with absolute certainty whether the participants deviated from the established definitions in their mind during the interview and there could be further terms that were perceived differently. The aspect of internal validity is of concern when causal relations are examined [43]. More specific, a study with a high internal validity accounts for confounding variables that could have unnoticed effects on the outcome. To prevent these confounding variables, the interviewees were asked about their previous project experience, their current project and their role in it. These details prevent that the requirements of the automation tool are heavily based on the desires of a specific role or the circumstances of a specific project. However, the relationship between interviewer and interviewee could undermine the internal validity, because the extent and detailedness of the answers of the interviewee might be higher when a mutual friendship exists. The external validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case [43]. The small sample size and the fact that all participants were volunteers do not contribute to a satisfactory external validity, because the general population may not be represented very well. It could also be that the machine learning expertise in the department is focused on specific approaches that work well for their relevant domain specific problems. Then the findings might not be applicable to other areas of ML.

4.6. Requirements extraction and prioritization

In a joint discussion with the industrial supervisor prior to the start of this master thesis, a high relevance for an automatic creation of the infrastructure and monitoring with real-time user data in the department was already suspected. The results of the interviews confirmed this assumption. When asked whether a tool for the automatic creation of the infrastructure would be useful (8b), seven interviewees answered that it would reduce effort, save time, improve reproducibility and reduce the error rate. The same question regarding monitoring (18b) was answered by nine interviewees, who said that changes in user behavior, data distribution and model quality can be easier identified. In order to derive requirements for an infrastructure and an automation tool from the results of the interviews, the evaluation results were discussed

Influencing factor	EG	EI	ED	EM
Literature search		RI1 RI3 RI5	RD2	RM1 RM2
Supervisor		RI2 RI4	RD1	RM3
10a		RI1 RI3		
16a			RD2 RD3	
9a	RG1			
7b		RI5		
25a		RI5		
21b				RM1
23b				RM1
19b				RM2

Table 4.3.: Influence factors for each of the requirements grouped by epic and ordered by descending influence.

together with the industrial supervisor and the answers to each question were examined to investigate whether a requirement could be derived. Additionally, the results of the literature search were examined. The extracted requirements were formulated as user stories and were grouped by epics. Four epics were created. One epic was created for general requirements (*EG*) regarding the use of an automation tool itself and three for requirements regarding infrastructure (*EI*), deployment (*ED*) and monitoring (*EM*).

The Table 4.3 shows which of the influence factors (literature search, supervisor and answers of the target-performance analysis) represent the source of the reasons for each of the requirements. The answers of the target-performance analysis are represented by alphanumerical identifiers and can be looked up in Tables B.1, B.2 and B.3 in the appendix. Regarding the influence factors, the literature search had the highest impact, followed by the supervisor and then the answers of interviewees. Among all questions, the questions 10a and 16a were particularly helpful for the identification of requirements. Out of the answers to 31 questions of the target-performance analysis, the answers of eight questions had influence on the requirements. This does not mean that the answers to the other questions were not important, they just could not be taken into account due to the limited scope of the requirements. It could also be recognised that answers to questions that have examined the actual state in the department had influence on five requirements and that answers to questions that focused on the target state had influence on four requirements. The concrete reasons for each of the requirements are mentioned in the following description of all epics.

The epic *EG* only contains the user story *RG1* that describes the purpose of using an automation tool. It is shown in Table 4.4. The reasons for the need of automation regarding the infrastructure and deployment were extracted out of the answers of question 8b and have already been explained in the previous paragraph and were considered important. One interviewee also mentioned that automation is only possible with appropriate know-how (9a). The

ID	User Story	Acceptance criteria	Priority
RG1	As a developer, I want to use an automation mechanism to reduce effort, save time, guarantee reproducibility and prevent errors.	- The tool can be operated by interacting with a graphical user interface	HIGHEST

Table 4.4.: User stories for the epic *EG* (general)

creation of an automation tool therefore also makes a contribution in this respect. Thus, the user story *RG1* was given top priority. To simplify the operation, the use of a graphical user interface was set as an acceptance criterium.

The epic *EI* contains user stories related to the configuration and creation of the training and application infrastructure, which are shown in Table 4.5. According to the supervisor, the creation of the application and the training infrastructure in *RI2* and *RI4* is both a frequent and important use case. However, the application infrastructure must be created more often, that's why the user stories *RI1* and *RI2* are therefore given highest priority. In *RI1* the developer configures the infrastructure with relevant parameters, such as the scaling threshold. The scaling threshold represents the value of the workload of the infrastructure at which scaling is initiated. According to the answers of question 10a, scalability is the most frequent non-functional requirement. In the literature, the practice of setting limits for a specific metric was frequently described and recommended. These reasons contribute to the priority score of

ID	User Story	Acceptance criteria	Priority
RI1	As a developer I want to make settings for the application infrastructure (e.g. scaling threshold) to provide parameters for its creation.	- Input values can be entered by the developer	HIGHEST
RI2	As a developer I want to create the application infrastructure to be able to deploy the application.	- The components of the application infrastructure are created	HIGHEST
RI3	As a developer I want to make settings for the training infrastructure (e.g. scaling threshold, training data location) to provide parameters for its creation.	- Input values can be entered by the developer	HIGH
RI4	As a developer I want to create the training infrastructure to be able to train the ML model.	- The components of the training infrastructure are created	HIGH
RI5	As a developer, I want to use automatic resource scaling to avoid the overload of the training or application infrastructure.	- The application infrastructure scales automatically when the load exceeds or falls below the scaling threshold	HIGH

Table 4.5.: User stories for the epic *EI* (infrastructure)

the user stories *RI1* and *RI3* and *RI5*. *RI5* referrs to the automatic scaling of the training and application infrastructure. The automatic scaling allows to meet the requirements of GPU and RAM, which were requested in the answers of questions 7b. Scalability is also particularly important in a production application (25a). Metrics for measuring application performance were also often mentioned in literature research. Since satisfactory application performance is not possible without a sufficiently powerful infrastructure, the user story *RI5* was given high priority.

ID	User Story	Acceptance criteria	Priority
RD1	As a developer, I want to make the ML application available to users to allow their access via HTTP.	- The uses are able to access the application via HTTP	HIGHEST
RD2	As a developer, I want to trigger and execute a Canary Deployment to provide the basis for monitoring with real-time user data	- Another instance of the application is deployed - A defined amount of user traffic is forwarded to the new application instance	HIGHEST
RD3	As a developer I want to trigger and execute a Blue-Green deployment to check the quality of the new application version	- Another instance of the application with all of its dependencies is deployed in a new environment (e.g. a virtual private cloud) - The complete user traffic is forwarded to the new application environment	MEDIUM

Table 4.6.: User stories for the epic *ED* (deployment)

Epic *ED* contains user stories that describe the deployment of the ML application, which are shown in Table 4.6. In *RD1* the ML application is made available to the users via HTTP. This user story is rated with highest priority because the application needs to be accessible to users, according to the industrial supervisor. In *RD2* the usage of a canary deployment is requested. A canary deployment was not yet used in any of the projects of the interviewees, but it was found to be useful by four interviewees (16a). In the literature search, canary deployment was the most frequent deployment method. Because of these reasons, *RD2* is rated with the highest priority. Two interviewees mentioned that they already performed a blue-green deployment (16a). As this method was not yet researched in the literature and the industrial supervisor could not give an estimate about the effort that it takes to implement, the priority of *RD3* was set to medium. The implementation of an A/B testing was discussed. It was decided not to further extend the scope of the requirements regarding the deployment, since the implementation of a canary deployment would be sufficient for the industrial supervisor. This is also the reason why shadowing was not further pursued.

The epic *EM* contains user stories that describe activities that are relevant for monitoring the ML application, model or the underlying infrastructure. They are shown in in Table 4.7. According to the answers of question 22b, monitoring is helpful for error reproducibility and analysis. The timeliness of the data has to be ensured and the CPU load should be monitored (21b). All of these metrics can also be found in the results of the literature search. At this point the inclusion of further metrics from the results of the literature search was discussed. It was decided not to include any more metrics as requirements, because the mentioned metrics already

ID	User Story	Acceptance criteria	Priority
RM1	As a developer, I want to be able to view metrics of an ML application and ML model on a dashboard to check their functionality. Metrics include CPU and RAM consumption, error rates, and age of the ML application and model.	- A dashboard is created with metrics regarding CPU/RAM usage, age and error rate of the model	HIGHEST
RM2	As a developer, I want to receive a message if the workload on the application infrastructure exceeds the scaling threshold to be notified.	- The developer receives a message per mail if the application infrastructure exceeds the scaling threshold	LOW
RM3	As a developer, I want to be able to see the current resource utilization, such as CPU/RAM load, of the application infrastructure to check it.	- A dashboard is created with workload information of the application infrastructure	LOW

Table 4.7.: User stories for the epic *EM* (monitoring)

provide a good basis for monitoring and the scope was considered sufficient by the supervisor. Due to the importance of the mentioned metrics *RM1* was rated with highest priority. As an acceptance criterium, the usage of a dashboard is expected, as it is a good means for visualization (23b). According to the results of the literature research and the interviews, especially question 19b, the sending of messages when thresholds are exceeded is a useful practice. As the scaling of the application infrastructure should work automatically, the user story *RM2* was set to low priority, because it is only a control function. This also applies to *RM3*. Since the scaling of the infrastructure is automatic, the view of the utilization values is useful for validation according to the supervisor, but it is not of high priority.

In summary, the ideas for implementing an infrastructure and an automation tool were in retrospect well reflected by the results of the interviews. Among the methods identified for monitoring the ML application and the model, a canary deployment was included as a requirement. A/B testing and shadowing were not included as a requirement in order not to exceed the scope. The metrics from the literature research partly overlapped with the desired ones from the interviews. Here, a small selection of the metrics was chosen, as the supervisor felt that they were sufficient. Many findings that emerged from the interviews were not included as requirements. One example is the fine-grained monitoring of calls to the ML application to find performance bottlenecks (18b) or the solution of problems related to the management of several AWS accounts simultaneously (9b). A limit had to be drawn so that, on the one hand, sufficient requirements for infrastructure and automation tool were imposed and, on the other hand, sufficient time was left for code quality assurance and testing.

5. Design, implementation and quality assurance

This chapter describes the design, implementation and quality assurance of the automation tool, which implements the requirements that were identified in the previous chapter.

5.1. Design and implementation

The design and implementation of the automation tool is covered in this section. First, a general overview over the automation tool is given, important architectural decisions are presented and the general implementation is described. Afterwards, the design and implementation of the core functionalities of the automation tool are explained.

Overview

The goal of the automation tool is to implement the requirements that were collected during the requirement analysis in Chapter 3. For this purpose a console application was developed in Python. The user is expected to be a developer in the area of machine learning, because the average interviewee that took part in the interviews for the requirement analysis was an engineer in this field.

Core functionalities

The automation tool provides five options to the user that represent its core functionalities. Table 5.1 shows which requirements are implemented by which core functionalities. The first core functionality is to create an application infrastructure for a given ML application. The application infrastructure is representative for all resources which are necessary to enable users the access to the ML application via the internet. The second core functionality is the creation of a training infrastructure. The training infrastructure is representative of all resources that are needed to train an ML model. The third and fourth core functionality enable the execution of a canary and blue green deployment. Both deployment types replace a current ML application version with a new one. While a canary deployment gradually redirects more and more users to the new ML application version, a blue green deployment instead redirects all users at once. The fifth core functionality allows to view monitoring metrics of an ML application and its infrastructure through the use of a dashboard. All five core functionalities are based on the orchestration of specific AWS (Amazon Web Services) services. These services are then used to take over different tasks, such as storing data, executing containers or sending notifications. AWS is used as a service provider, because of an ongoing partnership with the department of the company.

Core functionality	Covered requirements
Create application infrastructure	RI1
	RI2
	RI5
	RM2
Create training infrastructure	RI3
	RI4
	RI5
Execute blue green deployment	RD1
	RD2
	RD3
View monitoring metrics	RM1
	RM3

Table 5.1.: Mapping of core functionalities to requirements.

User interaction

A graphical user interface for the automation tool was not created, because a console application can be used more easily within automation scripts and furthermore provides better support for systems with no graphics card. When the start screen in Figure 5.1 is visible, the developer has the possibility to use the arrow keys to switch back and forth between the menu entries and to confirm the selection with the enter key. After entering the required settings for the selected functionality, the automation tool starts to execute an automation procedure, while simultaneously displaying the status of the execution progress in the console. When the execution is completed, the developer can extract relevant information from the console output. The automation tool is exited like any other console application by pressing the control key and "c" at the same time.

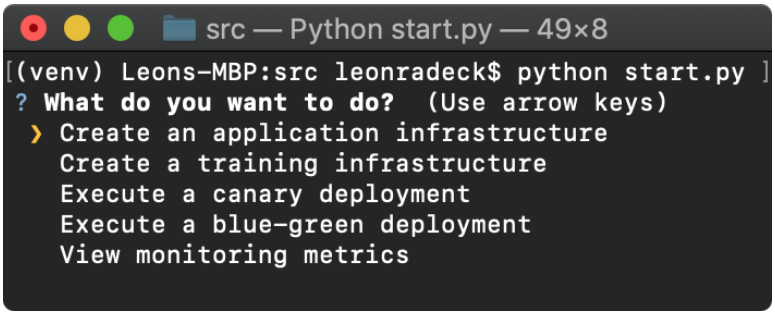


Figure 5.1.: Screenshot of the start screen of the automation tool.

Architectural decisions

Various architectural decisions had to be made regarding the implementation of the automation tool. One decision revolved around the selection of the programming paradigm. Here, object-oriented programming was compared to procedural programming. One of the main differences between object-oriented programming (OOP) and procedural programming is that the focus of procedural programming lies on dividing the programming task into a collection of variables and methods, while the focus of OOP is to decompose the programming task into objects that encapsulate variables and methods. Compared to procedural programming, which takes up less memory and allows to re-use the same piece of code at different places in the program without a detour over a class, OOP benefits from polymorphism and data encapsulation. Since the core functionalities of the automation tool are fundamentally based on polymorphism, the object-oriented approach was preferred over the procedural one. The decision knowledge visualization is shown in Figure 5.2 and was produced by the JIRA plugin ConDec [33]. It can be seen that the decision is displayed as the root of the graph. The next node below represents a problem that is associated with the decision. In this case, the problem is which programming paradigm should be used. The two alternatives that were compared, are shown in orange. The arguments for each of the alternatives are shown in green. For this problem, only arguments are used that support the alternatives.

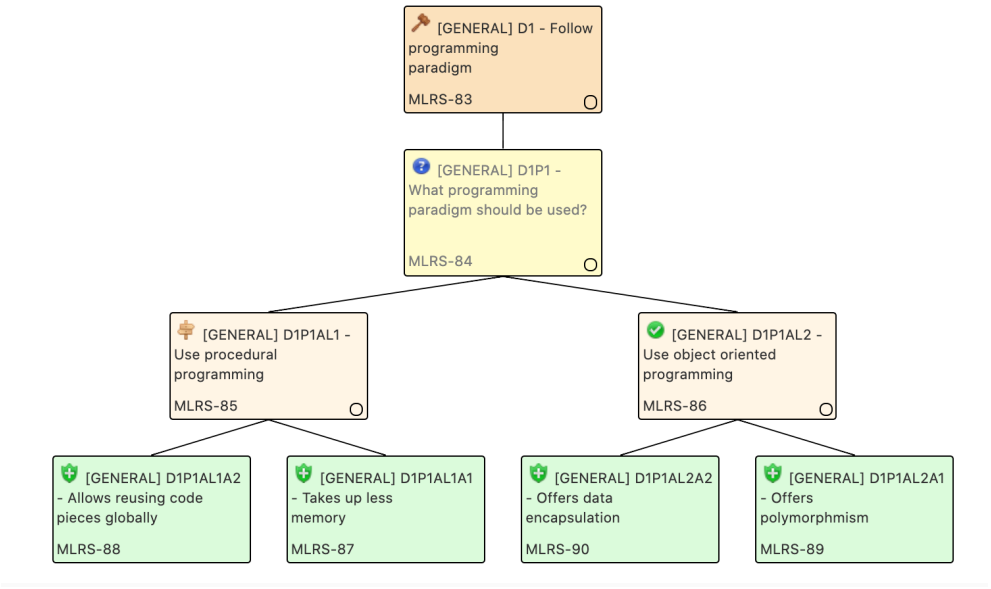


Figure 5.2.: ConDec decision knowledge visualization of decision about programming paradigm.

For the generation of resources, an IaC approach is used. IaC (Infrastructure as Code) means that all required resources and their configuration are defined in form of text files. On the basis of these text files, the resources and their configuration can be generated automatically with the help of dedicated tools. The tools CloudFormation [7] and TerraForm [28] were investigated in this regard. CloudFormation is natively offered by AWS and TerraForm is an OpenSource IaC tool. The decision was made on AWS CloudFormation because, unlike TerraForm, it offers the possibility to group the generated resources in so-called “stacks”, which greatly simplify collective resource deletion, as well as improvement of their traceability in the cloud.

General implementation

This section describes the general implementation of the automation tool. It first describes the `Stack` class and then all other classes.

A stack in the sense of CloudFormation is a collection of resources which is defined by a template file containing all resource definitions. Attributes of the resources can be dynamically initialized with values by the use of stack parameters, which makes it possible to configure the resources based on external input. The template file also contains an output section that defines a list of specific attributes of resources that should be accessible after the stack creation. Using OOP, the concept of a stack can be implemented by the abstract class *Stack*. The corresponding class diagram can be seen in Figure 5.3. The already mentioned parameters and outputs of the stack are modeled by the classes *StackParameters* and *StackOutputs*. The definitions of these classes are shown in the complete class diagram C.1 in the appendix. Further attributes of the *Stack* are its `_region` and its `_stack_name`. The region of the stack corresponds to the geographic location of the stack and the stack name is the identifier of the stack in AWS. The `_template_file_name` contains the name of the template file of the stack. In addition to the `_stack_parameters`, there is also the attribute `_action_parameters`, which encapsulates all parameters that are not needed for the stack creation, but still important for the implementation of the respective core functionality. For example, the training data file for the training of the ML model is an action parameter, because it is uploaded after the training infrastructure stack has been created. The last attribute of the stack is the `_command` that is necessary for the stack creation. When a stack object is instantiated via the constructor, the method `deploy()` executes the command to create the stack in AWS. The abstract methods _before_instantiate(), _after_instantiate() and _before_delete() are implemented by the respective concrete subtypes of *Stack* and allow the execution of code at certain points of the stacksĺife cycle. The other methods are not relevant for the understanding of this chapter and are thus not further described.

It is continued with the description of the remaining classes, leaving out those that are described in the next sections in context of the core functionalities. The class `Start` represents the entry point of the application. It is responsible for triggering the core functionalities that are selected by the developer. The class `Menu` contains all menu messages and handles the selection of menu

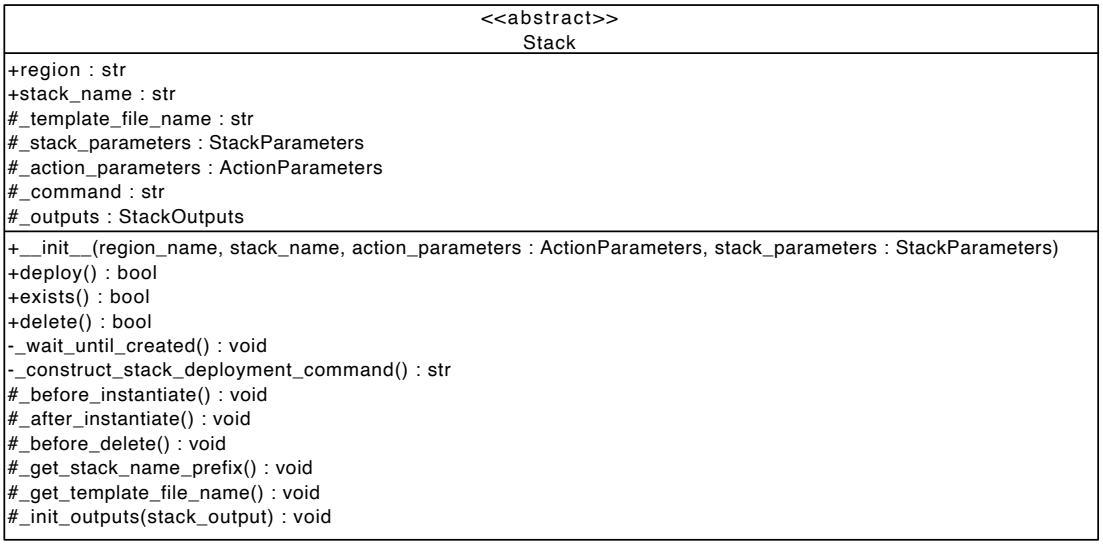


Figure 5.3.: Class diagram of the class *stack*.

items, It is responsible for receiving the developers input and provides default values whenever possible. The `Configuration` class contains constants that define the S3 bucket name that is necessary for the automation tool and the prefixes of the different stack types in AWS. The class `Serializable` allows to save a stack object to a file and read it back. This is helpful for testing. The `ShellExecutor` class is able to execute commands in the console. It is the superclass of the `StackHelper` class and the `DockerHelper` class. The `StackHelper` class offers utility methods to create a stack object from a remote CloudFormation stack, the get the outputs from a remote stack and to create a an S3 bucket that is necessary because of CloudFormation. The `DockerHelper` class contains utility functions regarding Docker. It allows to login to a docker repository, to tag, untag, pull, upload, remove or inspect an image and to list all images. It is noted that a summary of the descriptions of all classes of the automation tool can be found in Table C.1.

Application infrastructure

The first core functionality of the automation tool, is to create an application infrastructure. The purpose of the application infrastructure is to make an ML application available to users, while enduring peak traffic loads and guarantee high availability.

Resources

The application infrastructure consists of the following resources.

ECR repository - An Elastic Container Registry (ECR) repository [8] contains the docker image that was specified by the developer.

VPC - A Virtual Private Cloud (VPC) [6] is used to logically isolate resources of the application infrastructure. Within the VPC, a private and a public subnet exist. Resources in a public subnet can be accessed over the internet, resources in a private subnet cannot.

Internet Gateway - An internet gateway is the logical connection between the VPC and the internet. If the VPC had no internet gateway, then resources in the VPC could not be accessed from the internet.

NAT Gateway - A Network Address Translation (NAT) gateway allows a resource, that is inside a private subnet to connect to other AWS services.

ECS Task - An Elastic Container Service (ECS) task runs docker containers in ECS [5]. ECS is a fully managed container orchestration service in AWS. An ECS Task requires a task definition that must contain the location of the docker image.

ECS Service - An ECS service allows to run a specified number of instances of an ECS task simultaneously in an ECS cluster.

ECS Cluster - An ECS cluster is a logical grouping of ECS tasks or services.

Load Balancer - The application load balancer distributes the load between several application instances.

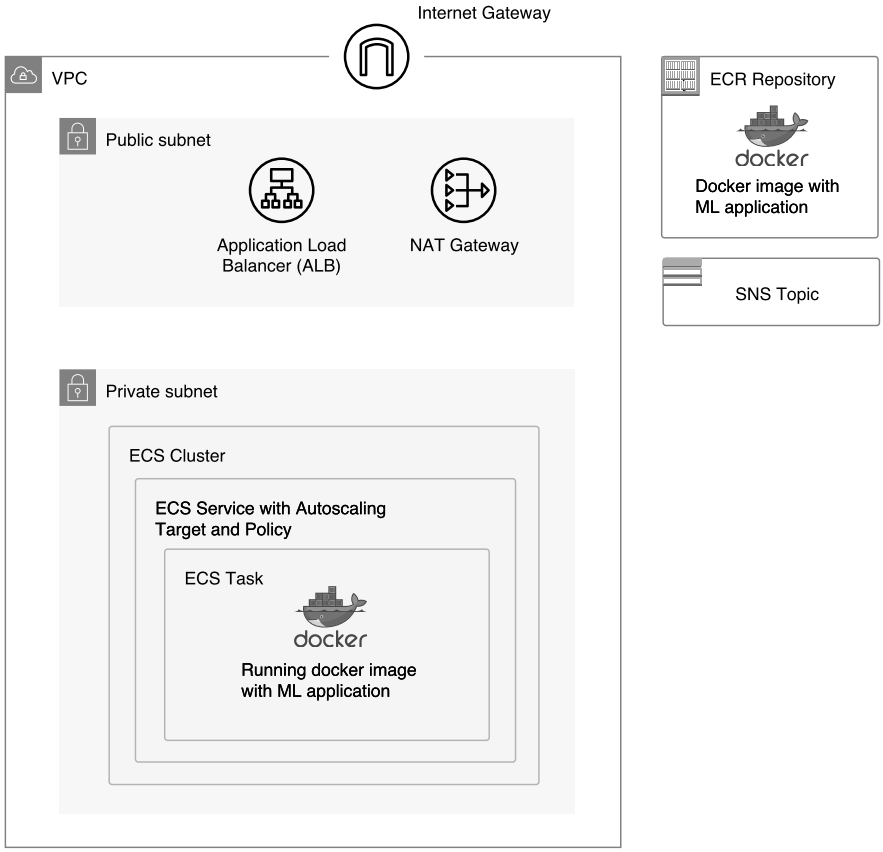


Figure 5.4.: Ressources of the application infrastructure.

Autoscaling Target - A resource that is associated with auto scaling functionality.

Autoscaling Policy - An autoscaling policy defines lower bound and upper bound metric values by which to scale in or scale out the autoscaling targets.

SNS Topic - A Simple Notification Service (SNS) topic is a logical access point that acts as a communication channel.

The Figure 5.4 shows all resources at a glance. The internet gateway receives the requests of the user of the ML application and forwards them to the public subnet of the VPC via the internet gateway. Within the public subnet, the requests reach the application load balancer. The load balancer checks the status of all ECS tasks by calling each of their health check paths. If the tasks responds with HTTP 200, they are marked as "healthy", if they do not respond or if they respond with another status code, they are marked as "unhealthy". The application load balancer decides to which of the healthy ECS tasks in the private subnet the requests should be forwarded based on the current load of the tasks. The corresponding task, which runs the docker image containing the ML application, receives the requests and sends their responses back to the NAT gateway in the public subnet. Afterwards the responses are forwarded back to the user over the internet gateway. It can be noticed, that the docker image is located in two places, the ECR repository and the ECS task. The difference is that the ECS task executes the docker image and the ECR repository only serves as storage space. When creating the application infrastructure, the ECR repository is first created, the docker image is uploaded to

it and then the ECS task is created with a reference to the docker image in the ECR repository. The last relevant resource of the application infrastructure is the SNS topic. The SNS topic is responsible for notifying the developer about scaling events. Every time a new ECS task is started or deleted, the SNS Topic receives an event and notifies all registered subscribers via email.

User flow

After choosing to create an application infrastructure in the start menu, the developer has to decide if the default region to create the infrastructure in is appropriate. If the developer denies, another region can be selected. Afterwards, a name for the project has to be entered, which will be used internally to name the infrastructure components. Next, the docker image must be selected and the docker image port must be entered. The health check path for the ML application has to be entered subsequently. The subsequent inputs all impact the scaling behavior of the application. The first input regarding the scalability is the minimum number of ECS tasks that are running concurrently. Assuming the number two is entered here, this means that two ECS tasks are created initially, for which the load balancer equally distributes the load. Afterwards, a maximum number of simultaneously running ECS tasks must be entered. Subsequently, the maximum CPU usage of an ECS task must be entered. Assuming the number 80 is entered here, that means that if any of the currently running ECS tasks CPU load exceeds 80%, then another ECS task is started, presuming the current number of ECS tasks is not already the maximum number. Finally, the developer provides an email address, that receives notifications when a new ECS task is started or deleted. The input of the settings is complete at this point. After entering the email address, the automation tool starts the creation of the application infrastructure in AWS. The developer is then provided with important information about the application infrastructure through the console output. This includes the link of the ML application and the names of various resources, for example the load balancer.

Implementation decisions and details

To enable the scalability and security of the application infrastructure, a suitable AWS service had to be found on which a dockerized application could be run and a dedicated load balancer had to be used to distribute the load between several application instances, while ensuring that every access to an application instance is always routed over the load balancer. As the service for the execution of a docker image in AWS, ECS was chosen. The alternatives EC2 (Elastic Compute Cloud) [4] and EKS (Elastic Kubernetes Service) [12] were also considered, but ECS promised less management effort and an on-demand pricing system. To enable scalability, the service ELB (Elastic Load Balancing) [13] was used, which allows to distribute user traffic to various ECS tasks using an application load balancer [10]. In order to protect the individual application instances from direct access over their IP addresses, which would circumvent the functionality of the load balancer, the service VPC [6] was used in combination with public and private subnets [9]. On code level, the application infrastructure is mapped by two concrete subtypes of the `Stack` class. The first subtype is called `EcrStack`. The only task of the `EcrStack` is to create the ECR repository and then upload the specified docker image to it. The `EcrActionParameters` only contain the docker image and the `EcrStackOutputs` consist of the ECR repository name and the identifier of the latest used docker image. Once the docker image is uploaded to the ECR repository, the task of the `EcrStack` is complete. Afterwards on object of the class `EcsStack` is instantiated. The `EcsStack` creates all remaining resources that form the application infrastructure shown in Figure 5.4. In particular, it creates the

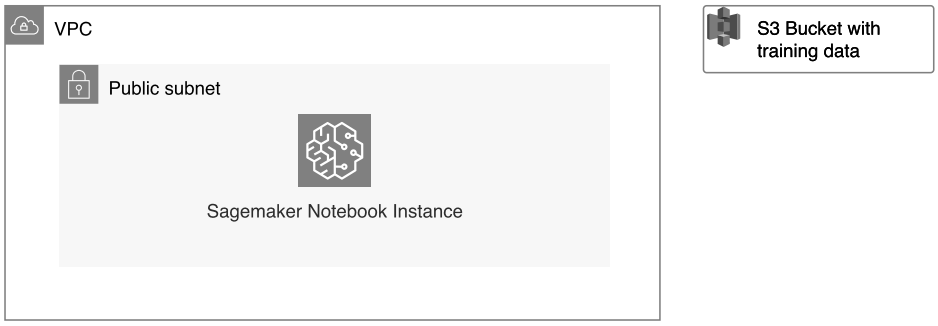


Figure 5.5.: Ressources of the training infrastructure.

configured number of ECS tasks from the existing docker image of the ECR repository. The `EcsStackParameters` contain the docker image, the docker image port, the minimum number of containers, the maximum number of container, the autoscaling threshold value, the subscriber email, the health check path, the ECR repository name and the name of the ECR stack. The `EcsStackOutputs` contain the ECS cluster name, the code deploy application name, the dashboard url, the dashboard name, the load balancer name, the name of the ECR repository, the name of the ECR stack, the endpoint of the ML application and some load balancer and security specific settings.

Training infrastructure

The second core functionality of the automation tool, is to create a training infrastructure. The purpose of the training infrastructure is to provide computing resources and data storage for the training of an ML model.

Resources

The training infrastructure consists of the following resources:

S3 Bucket - S3 is an object storage service of AWS. A bucket represents a location where objects can be stored.

Sagemaker Notebook Instance - A SageMaker notebook instance is a machine learning compute instance running the Jupyter Notebook App. Jupyter is a program which is used by data scientists to create so-called Jupyter notebooks. Jupyter notebooks are often used to program and train ML models.

VPC - A VPC is used to logically isolate the notebook instance. Within the VPC a public subnet exists.

The Figure 5.5 shows all resources in one view. The Sagemaker Notebook Instance is assigned to a public subnet which belongs to the VPC. Its computing power is configured by the developer through the console input. The S3 bucket contains the training data for the model training, which the developer specified during the settings of the training infrastructure. The data storage service that was used is AWS S3. It is the standard solution for storing files in AWS.

User flow

To create the training infrastructure, the developer first specifies the AWS region in which the training infrastructure will be created and enters a project name into the console. The developer then enters a path for a file with training data into the console. The training data will be used to train the ML model later. Subsequently, the developer chooses how much computing power is needed for the training of the model. For this purpose, the type of the AWS Sagemaker notebook instance must be selected. Once the settings have been entered, the automation tool begins to create the training infrastructure. After the creation of the infrastructure is finished, the developer accesses the Jupyter environment over an URL that can be extracted from the console output. The developer can then create a Jupyter notebook to start the training of an ML model. When the ML model training is complete, the developer can choose to save the model in the provided S3 bucket.

Implementation details

The functionality of the training infrastructure is represented by the class `TrainingStack` in the code. `TrainingStack`, like `EcrStack` and `EcsStack` inherits from the abstract class `Stack`. It further contains a method to upload the training data of the developer to the dedicated S3 bucket. This makes the training data available for the training on the notebook instance. The `EcrActionParameters` are initialized with the training file path and the `EcrStackParameters` contain the instance type of the notebook. The `EcrStackOutputs` contain the name and link of the S3 bucket, as well as the link to the jupyter notebook.

Canary and blue-green deployment

The third and fourth core functionality of the automation tool, enable the execution of a canary and blue green deployment. The purpose of these deployments is to exchange the currently running ML application version with a new one. The new application version can either be exposed to new users gradually in form of a canary deployment, or at once by using a blue green deployment.

Resources

The following resources are created before executing a canary or blue green deployment:

CodeDeploy Application - A container to hold the deployment group.

CodeDeploy Deployment Group - The basic configuration for a deployment.

New ECS Task - An updated version of the currently used ECS Task of the selected application infrastructure.

CodeDeploy Deployment - Process that deploys the new ECS task based on the task definition.

A CodeDeploy application and deployment group are created to encapsulate the deployment

configuration. Inside the deployment group, the deployment type is defined, which is either a canary or blue green deployment. Furthermore, attributes of the load balancer are configured so that the traffic can be redirected to the new ECS Task after it has been created. The new ECS task contains the docker image which is provided by the developer and it is created by executing the CodeDeploy deployment.

User flow

The steps necessary for the developer to execute a canary or blue green deployment are as follows. The developer first confirms if the default region should be used to search for the existing application infrastructures. If the developer denies, another region can be selected. Next, the developer selects the desired application infrastructure, which is be the basis for the canary or blue green deployment. The developer then selects the new docker image that contains the new ML application version. The docker image port and health check path have to be entered afterwards. When these inputs are done, the execution of the corresponding deployment type starts. In the console the developer can then open up a link to a page with information about the deployment.

Implementation details

In the code, the functionality of the deployment is provided with the class `DeploymentMixin`. Mixins are used in object-oriented programming to enrich an existing class with properties and methods. In this case, the class `EcsStack`, which represents the application infrastructure, is extended with functionality from the `DeploymentMixin` class through inheritance. Since the class `EcsStack` already inherits from the abstract class `Stack`, this is its second superclass. Within Python, multiple inheritance is possible and especially useful in this case. The advantage of this approach is, that within the `DeploymentMixin` class the attributes of the `EcsStack` can be easily accessed and used to achieve the required functionality. The `DeploymentActionParameters` contain the docker image and its port.

Monitoring metrics

The fifth core functionality of the automation tool is the presentation of monitoring metrics regarding the application infrastructure, the ML application and ML model. The only resource that has to be created within AWS is a dashboard for the metrics. For the developer, the access to the dashboard is easy. Within the automation tool, the application infrastructure is selected and the link to the dashboard appears in the console.

Implementation decisions and details

AWS CloudWatch was chosen as a service to display metrics through the use of a dashboard. The alternative would have been to program a web application with a dashboard in the frontend that displays the desired metrics. Although the flexibility of the presentation options would be higher in this case, the development effort would be considerable. Since the AWS SDK also offers the possibility to send metric values to the dashboard via an API, it was decided to use the combined functionality of a CloudWatch dashboard and the AWS SDK. The CloudWatch

dashboard is configured to show the CPU and RAM consumption of the application infrastructure and ML application, as well as the error rate of the responses that the ML application produces. It is noted, that the metrics for the error rate of the ML model, for its age and the age of the ML application require manipulation of the corresponding software code. Similar to the deployment functionality, the monitoring functionality is represented as a mixin which extends the class `EcsStack`. The mixin `MonitoringMixin` contains methods to construct metrics and to add them to the dashboard. The mixin has full access to the attributes of the `EcsStack` which facilitates the metric creation.

5.2. Quality assurance

To ensure the quality of the automation tool, a combination of testing, static code analysis and code documentation was used. Component and system tests were created to test individual pieces of software and to determine if the entire automation tool works under realistic inputs. The test coverage is 93%. It was made sure that the tests can be executed in parallel, which greatly reduces their runtimes.

A static code analysis was performed to gain insight into metrics such as LOC (lines of code) and average cyclomatic complexity. In Figure 5.6 the LOC metric for each python class and template file can be seen. The total number of code lines is 2138, where 1308 lines are python code and 830 lines are template code. The average cyclomatic complexity of a python class is A 1.5, which means it is of low risk and high maintainability. There are 143 python methods in the source code which themselves are associated with 25 classes. The metrics were collected with a command line tool called radon [1]. During programming, the coding convention PEP8 is followed and Flake8, a python linter, is used to highlight errors and spelling mistakes within the code. Code comments were added to classes and methods, listing the reason for their existence and their parameters.

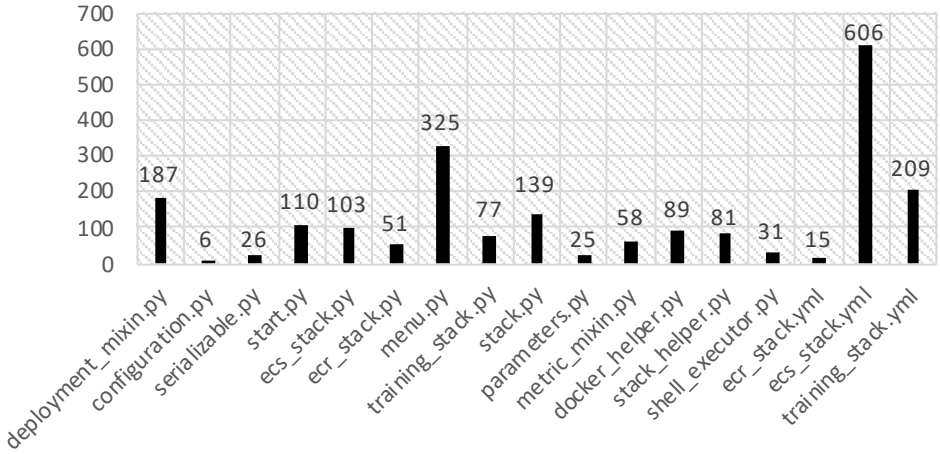


Figure 5.6.: LOC metric for each python file.

The unit tests are executed with each commit by the use of a pipeline which was created in Gitlab. The unit tests check if the action and stack parameters are constructed by variables of the right type. For example, it is tested whether numbers are entered when numeric inputs are

expected and whether special characters are denied. The correct serialization and deserialization of stack objects in the class `Serializable` is also checked. Here it is tested whether files are created in the right places and whether all attributes of the object can be saved and restored correctly. The methods of the helper classes are also tested for correct functionality. In the class `StackHelper` it is tested whether a CloudFormation stack can be successfully converted into the instance of a `Stack` class, whether all attributes are set correctly and if the output of a CloudFormation Stack can be read successfully. In the class `DockerHelper` the methods are checked for the case that the docker image does not exist or attributes like its tag are missing. The `ShellExecutor` class checks whether the method for executing a command produces valid results. Within the unit tests, method calls are mocked if necessary. This is sometimes required for API calls of the AWS SDK.

The integration tests are ran manually, because a permanent execution would cause significant

ID	Prerequisite	Description	Expected result
ST1	-	1. Choose „Create application infrastructure“ 2. Select the region (“eu-central-1”) 3. Enter a random alphanumeric project name 4. Select docker image („nginx:mainline-alpine”) 5. Enter docker port („80“) 6. Enter health chech path („/“) 7. Enter minimum number of containers (1) 8. Enter maximum number of containers (2) 9. Enter autoscaling value (0.1) 10. Enter email leon.radeck@web.de 11. Wait until application infrastructure is created 12. Execute a load test	• An email was sent with a notification of a starting ECS Task • The application is reachable under the corresponding URL
ST2	-	1. Choose “Create training infrastructure” 2. Select the region (“eu-central-1”) 3. Enter a random alphanumeric project name 4. Enter the path of the training data file (“./data.csv”) 5. Select the instance type (“ml.t2.medium”)	• The training data was uploaded to the S3 bucket • The training instance notebook is reachable over its URL
ST3	An application infrastructure exists	1. Choose “Execute canary deployment” 2. Select the region (“eu-central-1”) 3. Select the previously created application infrastructure 4. Select the docker image (“nginx:mainline-alpine”) 5. Enter the docker port (80)	• The old application version is replaced with the new one • The new application version is reachable over its URL
ST4	An application infrastructure exists	1. Choose “Execute blue green deployment” 2. Select the region (“eu-central-1”) 3. Select the previously created application infrastructure 4. Select the docker image (“nginx:mainline-alpine”) 5. Enter the docker port (80)	• The old application version is replaced with the new one • The new application version is reachable over its URL
ST5	An application infrastructure exists	1. Choose “Show monitoring metrics” 2. Select the region (“eu-central-1”) 3. Select the previously created application infrastructure	• The dashboard is reachable over its URL

Figure 5.7.: System tests.

costs in AWS. An overview of the system tests can be seen in Figure 5.7. *ST1* tests the creation of an application infrastructure in the default region with a random alphanumeric project name. The docker image "nginx:mainline-alpine" is used as an example application. The minimum number of containers is set to "1" and the maximum number to "2" so that the scaling can be tested. The autoscaling value is set to "0.1" so that even few requests will trigger the scaling of the application infrastructure. A load test is executed subsequently so that it can be checked whether an email was sent with a notification. The reachability of the application is tested additionally. *ST2* tests the creating of the training infrastructure. The project name is again a random alphanumeric string. Example training data is provided and as the instance type "ml.t2.medium" is chosen. It is tested whether the training data was uploaded to the S3 bucket and whether the instance notebook is reachable over its URL. *ST3* and *ST3* test the execution of a canary and blue green deployment. Both tests follow the same structure. An application infrastructure has to be created beforehand to execute the tests. The default region is used and the previously created application infrastructure is selected. The nginx docker image with port 80 is used as an example again. It is tested whether the old application version is replaced by the new one and whether the new application version is reachable over its URL. *ST5* tests if the monitoring metrics are reachable. First, the default region is used and the previously created application infrastructure is selected. Then it is tested whether the dashboard for the metrics is reachable over its URL.

6. Evaluation

This chapter describes how the evaluation of the automation tool was prepared, how it was executed and what results were obtained.

6.1. Preparation and execution of the evaluation

The goal of the evaluation is to answer the key evaluation questions in Table 6.1 to validate how good the automation tool matches the requirements that were identified in Chapter 2. *EQ1* and *EQ2* each aim to validate the configuration, creation and scaling of the application and training infrastructure. *EQ3* addresses the process of making available the ML application and the execution of a blue green deployment, whereas *EQ4* covers the monitoring metrics about the ML application, ML model and application infrastructure, as well as the automatic scaling notifications. The validation of the execution of a canary deployment is omitted because it hardly differs from that of a blue green deployment and the evaluation schedule was already time extensive.

ID	Requirements	Question
EQ1	RI1, RI1, RI5	Is the automation tool suitable for creating an application infrastructure?
EQ2	RI2, RI3, RI5	Is the automation tool suitable for creating a training infrastructure?
EQ3	RD1, RD3	Is the automation tool suitable for executing a blue green deployment?
EQ4	RM1, RM2, RM3	Is the automation tool suitable for viewing monitoring metrics?

Table 6.1.: Key evaluation questions.

In order to answer the evaluation questions, the evaluation guideline, which is shown in Figures D.1, D.2 and D.3 was developed. In the introduction of the evaluation guideline the participants are welcomed and thanks are given for their willingness to take part in the evaluation. Just like in the requirement analysis, the participants are told that their name will only appear pseudonymously in connection with the evaluation results. In the next section, the participants are briefed with general information about the automation tool. Here, the meaning of the inputs and outputs of the automation tool is explained and its functionality is described. In the following section, the participants perform predefined usage scenarios while operating the automation tool independently. The design of the usage scenarios is based on two considerations. First, all functionalities that fulfill the requirements have to be validated by the participants. In the evaluation guideline, the steps of the usage scenarios are therefore mapped to the corresponding requirements that they validate. For example, when the developer follows the instructions to check that the application infrastructure correctly scaled after a load test, the corresponding requirement *RI5* is noted next these instructions. Second, the instructions

for the usage scenarios are described in great detail, so that the participants are not dependent on external help. Although it would have been easier to identify usability problems with coarser instructions, the focus of the evaluation was to ensure that all usage scenarios were executed within the specified time frame. In the last section of the evaluation guideline, the participants answer a questionnaire that follows the TAM model [21] to measure the perceived ease-of-use ($PEOU$), perceived usefulness (PU) and behavioral intention (BI) of the functionalities of the automation tool. Perceived ease-of-use is defined as "the degree to which an individual believes that using a particular system would be free of physical and mental effort". Perceived usefulness is defined as "the degree to which an individual believes that using a particular system would enhance his or her job performance". Behavioral intention is defined as "the individual's subjective probability that he or she will perform a specified behavior". The questionnaire consists out of 39 questions, which have to be answered by the participants using one of the following options: "Strongly disagree", "Disagree", "Neutral", "Agree", "Strongly Agree". Each question focuses on one TAM criterium and can be mapped to a requirement and an evaluation question. Each requirement is addressed by a number of questions which each focus on either PU, $PEOU$ or BI. The questions which focus on PU ask whether the corresponding functionality is simple to perform or to understand. If the PU is high, the effort to use the automation tool would be low. Questions with the PU criterion aim to recognize the meaningfulness and the degree of coverage of the respective functionality. If the meaningfulness and task coverage is high, the automation tool would positively influence the work performance of the participant. Lastly, questions that address the BI of the participant ask if the functionality will be used in the future. A high BI would mean the participant is sure to continue to use the functionality of the automation tool and a low BI could mean that difficulties occured during its operation. The BI of the participant is especially important, because it is influenced by PU and $PEOU$. Due to the orientation of the questions according to the TAM model, the evaluation questions can be evaluated in a differentiated way according to the criteria PU, $PEOU$ and BI and the general feedback that can be given by the participants in form of free text.

For the organization of the evaluation, an email was sent to twelve employees of the department. Ten employees agreed to participate in the evaluation. The meetings were scheduled for 45 minutes. Since face-to-face meetings were not possible, the evaluation was conducted online via Microsoft Teams. In order to do this, it was made sure in advance that the technical possibility of sharing the screen control was working. Furthermore, it was checked whether the transmission quality was sufficient for an uninterrupted screen transfer. The questionnaire was provided to the participants over Google Forms. During the evaluation, the screen control was shared via Microsoft Teams and the participants had control over mouse and keyboard. The usage scenarios were documented as step by step instructions in a text file and could be read after the screen was shared. The participants switched back and forth between the window of the text file and the window of the automation tool during the execution of the usage scenarios.

6.2. Results

The results of the questionnaire are presented in this section. Out of the ten participants that took part in the evaluation, seven filled out the questionnaire. Sending a circular email to all participants as a reminder had no success to motivate the remaining three participants to fill out the questionnaire. In total, the participants strongly agreed to the questions in 46% of the cases, they agreed in 35%, they chose "Neutral" in 15% and "Disagree" in 4%. No participant strongly disagreed with any question. In the following the results regarding the evaluation questions *EQ1*, *EQ2*, *EQ3* and *EQ4* are presented. To find out which participant gave which answer and why, the complete result table can be found on pages 89-97 in the appendix.

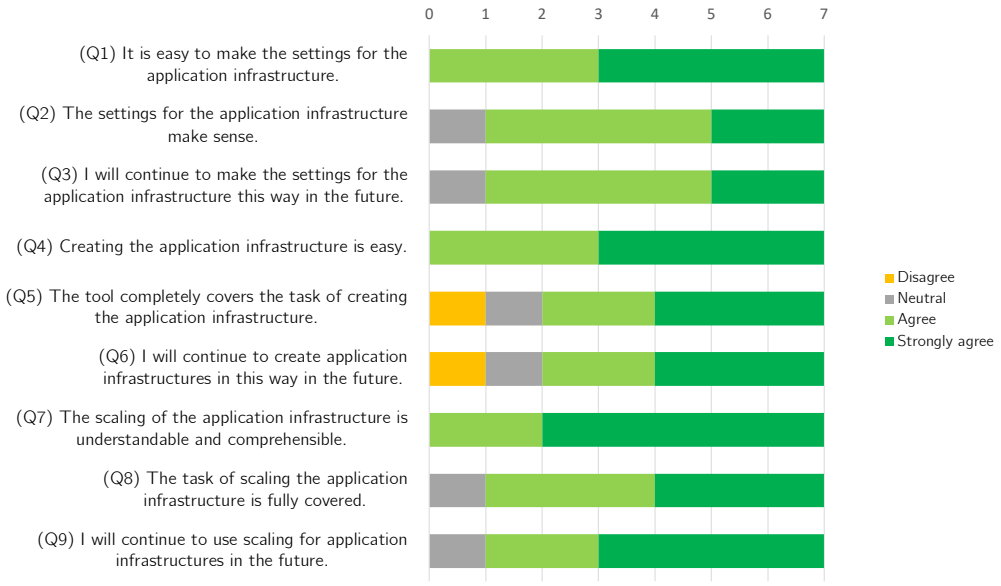


Figure 6.1.: Results of questions regarding *EQ1*

[*PEOU*] The answers of the questions *Q1*, *Q4* and *Q7* were all positive. Regarding the configuration of the application infrastructure, the participants stated that it would be *self-explanatory, clear and understandable*. The steps in the console would be *easy to understand* and *good default values are provided*. As feedback it was mentioned that *clicking on the menu items should not throw an error* and that *the settings may be more specific for some applications*. Concerning the creation of the application infrastructure, the participants said that it would be very easy again and that *the degree of automation would be very high*. They complimented that it would *save a lot of time effort compared to manually executing the steps in AWS*. A *better structure for the console output* was proposed to make the outputs more clear. The scaling of the application infrastructure was *understandable and clear* to all participants. It was added that *horizontal scalability cannot be used in all cases* though.

[*PU*] Among the questions *Q2*, *Q5* and *Q8* there were three neutral answers and one negative answer next to the positive answers. According to the results of *Q2* the settings of the application infrastructure were *easy to understand and clear* for all except one participant who chose neutral. The reason for the neutral answer was that the participant was not sure what would have happened when invalid inputs were entered. The reasons for the positive answers were, that *the settings cover most scenarios well* and that *most points are adjustable*. It was proposed to not only use CPU usage as a metric but also *the number of requests* and *RAM usage*. One participant mentioned that *every project has different requirements* and that it would be hard to find general settings for all projects. *Q5* asked about the coverage of the application infrastructure creation and had one neutral and one negative answer. The reason for the neutral answer was that the participant *misses the aspects of data integration, multi-user usage, versioning and security*. The participant who disagreed, said that *not all details of the application infrastructure are individually adjustable*. The remaining answers were all positive with no special reasons. *Q8* was rated positively by all but one participant. The reasons for the positive answers were, that the scaling would be *very clear* through the use of CloudWatch and emails and that an additional instance would be started when the CPU load is reached. All typical scenarios are covered and the scaling would be good for services. It was added that

disk space and *system failure* should be included as scaling metrics as well. One participant gave a neutral answer with the reason, that it would be unknown whether all scaling settings can be adjusted.

[*BI*] The results for the questions *Q3*, *Q6* and *Q9* were positive except three neutral answers and one negative. The reasons for the positive answers regarding the configuration of the infrastructure were, that it would be *simple*, *quick* and *repeatable*. However, it was emphasized that *the settings would have to be adjusted* according to the individual project requirements when used in the future. The necessary adaptation was also the reason for the neutral answer. The creation of the infrastructure was perceived as *quick*, *easy* and *uniform*. One neutral answer was given with the reason that setting up the application infrastructure would be not the task of the participant. The reason for the negative answer was, that more individual settings would be necessary. One participant liked the scaling of the application infrastructure, because it was *comfortable* and *useful for applications with fluctuating access rates*. The reason for the only neutral answer was that the *scaling depends on the system* and its requirements.

In the general feedback, the handling of the automation tool was complimented and it was pointed out, that the automation tool can *save a lot of effort*. It was proposed to *visualize the output in a more compressed way* and to make the settings more flexible.

EQ2

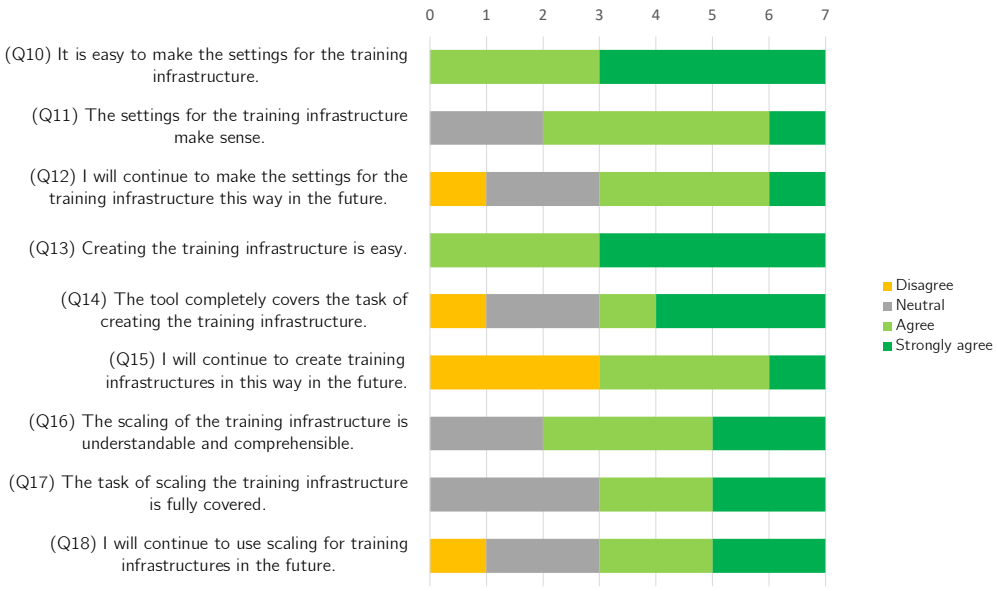


Figure 6.2.: Results of questions regarding *EQ2*

[*PEOU*] The questions *Q10*, *Q13* had only positive answers. The reasons for that were that the settings would be *self-explanatory*, the automation tool would give *detailed instructions* and it would be *easy* to create a training infrastructure. It was proposed to *add a retraining feature* to periodically execute the training and to *use MetaFlow instead of Sagemaker*. The results of *Q16* contained two neutral answers, because the *vertical scaling would be not evident* to the participants. The other answers were positive, because the scaling was perceived as *understandable* and *comprehensible*. One participant noted that it would be not clear what Sagemaker does with intermediate results.

[*PU*] The answers of the questions *Q11*, *Q14* and *Q17* were once negative and seven times neutral. The reasons for the neutral answers were that *no arbitrary training structures can be created*, *no versioning would be available* and *no training results could be reported*. It was also *unclear which libraries could be used for training* and that the *vertical scaling would be not evident*. The reason for the negative answer was that the creation of a training infrastructure would be not necessary in the current project. The positive answers stated that the settings of the training infrastructure would be *simple*, *sufficient* and *well explained*. Some participants used the same reasons for multiple answers, which were not listed here.

[*BI*] The questions *Q12*, *Q15* and *Q18* had the most negative answers in summary. The reasons for the five negative answers were, that it would be not necessary to use the automation tool in the current project, because the *training infrastructure already exists*, the usage of the training infrastructure would be *highly dependent on the data, algorithms and preprocessing*, it would not be the task of the participant to create the infrastructure and *horizontal scaling would be preferred over vertical scaling*. The four neutral answers were reasoned with a *missing possibility to connect data sources* for the training infrastructure and that the scaling would be either *not always necessary* or it would be *dependent on the scenario*.

In the general feedback, the automation tool was described as *very useful* and it was proposed that features for *versioning*, *user rights* and *optional scaling* should be added.

EQ3

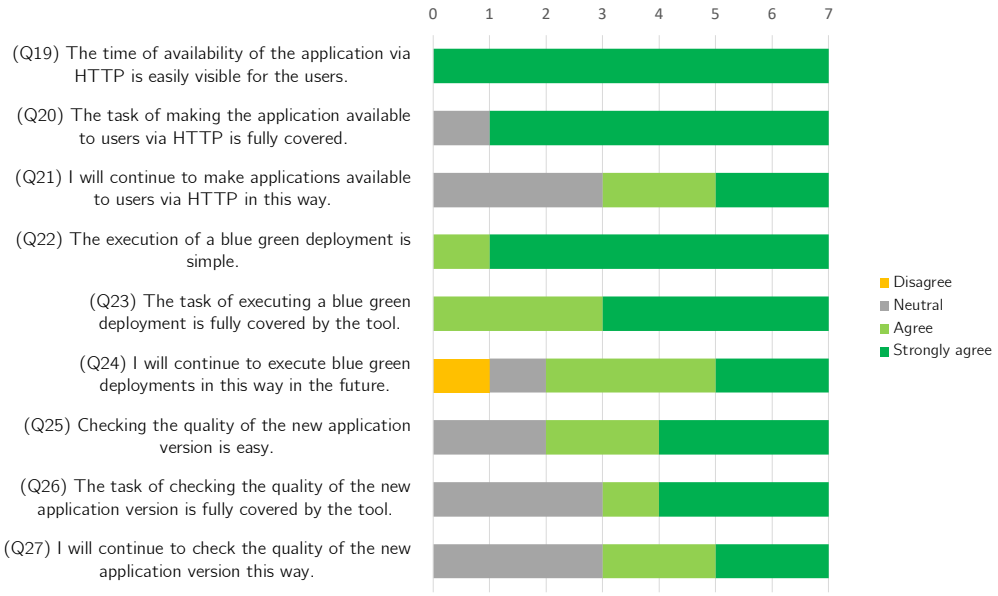


Figure 6.3.: Results of questions regarding *EQ3*

[*PEOU*] The answers of the questions *Q19*, *Q22* and *Q25* were positive except for two neutral answers. The reasons for the positive answers were as following. The availability of the ML application would be *indicated in the output* and it would be *clearly visible by the AWS GUI* through helpful visual dashboards. The execution of a blue green deployment would be *self-explanatory*, *understandable* and *simple*. Checking the quality of the new application version would be *easy* with the help of the metrics. One answer was neutral because it was unclear to the participant what was meant by “quality”. The other neutral answer was given because of missing experience.

[*PU*] The results of *Q20*, *Q23*, *Q26* consisted of four neutral answers and many positive ones. Regarding the positive answers, it was stated that the *deployment of the application would work properly* and it would always be *clear which application was running and when the new one would be made available*. One participant added that it would be even better to *integrate the task of making the application available into a CI/CD pipeline*. One neutral answer was given because of a lack of knowledge in the completeness of HTTP applications. Another participant noted that *tests and abort scenarios should be used* in combination with the blue green deployment. The remaining three neutral answers were given because of missing knowledge and because it would be *not sure whether the automation tool checks if the new application version was deployed*. Furthermore, it was added that *test scenarios would have to be considered in production deployments*.

[*BI*] The questions *Q21*, *Q24* and *Q27* were answered with one negative and seven neutral answers, next to the positive ones. In the positive answers the participants mentioned that the task of making the application available would be *definitely covered* and that they consider to continue to use the automation tool. The three neutral participants said it would *depend on the application and requirements*. One participant disagreed to continue to use the blue green deployment because the task would be *covered by a CI/CD pipeline* in the project. It was added that the automation tool could be used in this context. One participant was neutral because of missing experience. The remaining participants agreed to use the blue green deployment in the future. One added that it would further be nice to have an *overview over all deployments of all services*. Checking the quality of the new application will be done by four participants in the future. The other three were neutral, because of missing experience and because it would depend on the project.

In the general feedback the participants said that the automation tool would be *very easy to use*, that it would be *limited to the most necessary functions* and that a blue green deployment could be *easily executed*. It was added that the automation tool *would have to be extended for more complex systems*.

EQ4

[*PEOU*] The questions *Q28*, *31* and *Q34* only contain one neutral answer next to the positive ones. The reasons for the positive answers were that the metrics would be *easy to understand* and *very clear*. It was proposed to *add other metrics too, to be used for management reporting*. One neutral participant mentioned that the metrics would be clear but *not totally understandable for someone who has no idea of machine learning*. Regarding *Q37*, all participants agreed, because *the notifications would be helpful and were written in an understandable way*. One participant added that *displaying the message on a dashboard would be even better*, because emails would sometimes be not read.

[*PU*] The answers of questions *Q29*, *Q32* and *Q35* were rated three times neutral and once negative, next to the positive ones. The reasons for the positive answers were that metrics would be *mandatory to monitor ML applications*, that it would be *easy to understand* the ML model with the existing metrics and that all informations about the application infrastructure would be *visible at a glance*. It was proposed to *add additional metrics* depending on the model type. The neutral participants stated that, regarding the ML application, it *depends on the hypothesis* that is investigated. *Every project would have to create its own KPIs manually*, but the existing metrics would be a good starting point. Regarding the ML model, one participant could not fully agree to the question and another one had problems to differentiate the meaning of the metrics. According to the negative answer, the more complicated a model is, the more difficult it would be to understand it. The metrics would not help here, because *each model would be different* and has *different properties*. Regarding *Q38* there was one neutral answer,

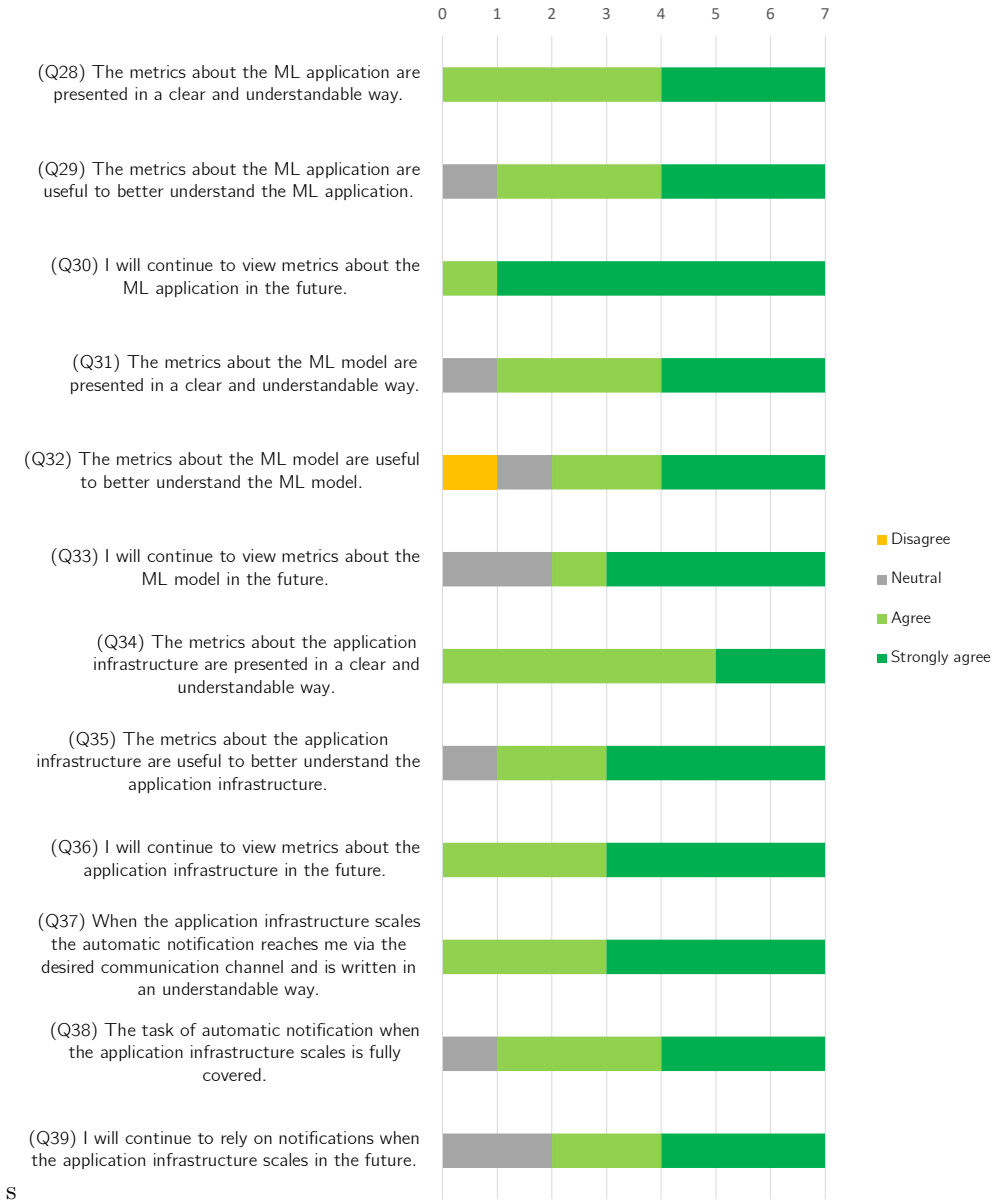


Figure 6.4.: Results of questions regarding *EQ4*

because a dashboard would be preferred over an email notification. The other participants agreed, that the task of automatic notification would be *fully covered*. One mentioned that a notification about the *accuracy of the trained model* would be important too.

[*BI*] Among the questions *Q30*, *Q33* and *Q36* there were only two neutral answers besides the positives ones. The participants agreed to continue to view metrics about the ML application, ML model and the application infrastructure, because *without monitoring the quality could not be checked*, which would be necessary during operation. The neutral participants were either not sure or said it would be *dependent on the ML model*. The results of question *Q39* were positive except for two neutral answers. The neutral participants mentioned that notifications would be interesting for the company and that *a dashboard could be used alternatively to the emails*. Other participants pointed out that there should always be notifications and that they

would be very useful.

In the general feedback the participants stated that the automation tool would be very *helpful* and *efficient*. For one participant, the questions regarding the monitoring metrics were *too extensive* and they should have been *more precise* and *summarized*. Another participant mentioned that the metrics about the infrastructure *could be used as a template*, while the metrics for the ML model would have to be adjusted by hand. The importance of the metrics was emphasized by another participant.

It is noted that in this section all free text answers of the participants were taken up except for some repetitive, inexplicit or incomprehensible answers which have been omitted to improve the reading flow.

6.3. Discussion

The conduction of the evaluation of the automation tool provided interesting insights into the perspectives of the participants. It is now discussed whether the change and extension requests of the participants should be included into the functionality of the automation tool. The change requests are discussed in descending order of relevance.

[**Application infrastructure**] It has been pointed out that the application infrastructure settings cover most common conditions, but would be too inflexible for more complex projects. This comment is understandable. In order to increase the flexibility of the setting, the individual characteristics of the various projects must be identified and the current rigid configuration structure must be loosened up. This adaptation would be demanding and require a lot of programming. For the broad use of the automation tool, however, it is necessary. With regard to the scaling of the application infrastructure, it was suggested to use other metrics, such as the request rate disk storage or system failures to trigger scaling activities. This suggestion is probably easy to implement, since AWS offers various metrics by default. Therefore the cost/benefit ratio would be relatively good and the functionality should be included in my opinion. It was noted that an error occurs during the creation of the application infrastructure, when the developer clicks on menu items of the automation tool. The remark is completely understandable and the bug should be fixed in any case. The improvement of the structure of the console output is obvious. The output would benefit from a clearer tabular form, which is easy to implement. It was also suggested to create an application infrastructure via Kubernetes. In this respect, the current architecture offers potential, since the use of the technology also allows the launching of Kubernetes pods. However, the changeover would involve a great deal of effort and it is advised to first reflect whether the goals cannot be achieved with the current approach through AWS ECS. Some comments could not really be understood. For example, it was incomprehensible what exactly the meaning of multi-user usage or versioning would be. It was also suggested to create an application infrastructure via Kubernetes. In this respect, the current architecture offers potential, since the use of the technology also allows the launching of Kubernetes pods. However, the changeover would involve a great deal of effort and it is advised to first reflect whether the goals cannot be achieved with the current approach through AWS ECS.

[**Training infrastructure**] With regard to the training infrastructure, a versioning of the training data makes sense, because the role of data is especially important and hard to handle in machine learning. Here, the versioning function of AWS S3 could be used to begin with. The possibility to vertically scale the training infrastructure should be visible more prominently in the settings. Maybe it was not obvious that different CPU and RAM specifications can

be generated by choosing the notebook instance types or the term “scaling” was interpreted differently. An optional scaling feature would be very easy to implement and should be integrated in the automation tool. The generation of result reports regarding the model training is considered useful. It would be necessary to research the possibilities that Sagemaker offers in this regard. The mentioned periodic retraining feature could be a helpful functionality to avoid having to start the training manually. In this case, it would be necessary to communicate with the developers from the department to determine whether this feature is desired by other parties too. According to one participant, the use of clusters and multiple instances would be desirable. Since I myself come from the software engineering field, I cannot judge with certainty whether this feature is absolutely necessary. It could be that for larger models a parallelisation of the model training is important. The training infrastructure would then have to be adapted accordingly. Using MetaFlow as an alternative to Sagemaker cannot be judged by me, as I have no experience with MetaFlow.

[**Blue green deployment**] Currently, the developer can only check the metrics manually. Automatic analysis of the metrics in combination with a rollback functionality is desirable and essential for productive use. In an internal meeting of the department, this functionality was already presented as a proof of concept and it should be included in the automation tool in the future. The combination of tests with a blue green deployment is particularly important in this context. The integration of the blue green deployment functionality of the automation tool into a CI/CD pipeline makes total sense. Since the automation tool is a console application, a blue green deployment could be started by passing parameters to the application from a script. With regard to the mentioned overview of all deployed services, it would be necessary to communicate again with the participant to have a clear understanding about this feature.

[**Monitoring metrics**] In terms of scaling notifications, a dashboard was requested. For this, the existing dashboard of the monitoring metrics could be extended by the scaling metric. This feature would be relatively easy to implement and should therefore be included. The use of multiple metrics in the context of management reporting makes sense. The relevance of metrics is strong enough to justify this request. The programming effort is difficult to estimate, as it depends on the conditions of the reporting.

6.4. Lessons learned

The execution of the online evaluation with the help of screen sharing turned out to be less problematic than initially thought. Microsoft Teams worked satisfactorily, even if there were some minor technical problems. The keyboard layout did not allow any underscores or dashes to be entered by the participants and the scrolling speed was hard to control. These circumstances made remote control of the computer more difficult, but did not seriously affect the progress of the evaluation. The evaluation of the scaling behavior of the application infrastructure was unproblematic in all but one case. In this case, the requests sent to simulate user traffic were not sufficient to exceed the scaling threshold of the application infrastructure. In this point, the evaluation took a little longer because the load test had to be repeated with more requests. All participants showed great interest in the automation tool and the atmosphere during the evaluations was not stressful or tense. The use of Google Forms as a platform for the distribution and completion of the questionnaire turned out to be user-friendly and unproblematic. The questionnaire could be shared via a link and the results of the questions were available as a tabular file. Although only 70% of the participants filled out the questionnaire, with 39 questions it was relatively extensive. Furthermore, all fields were mandatory fields, which might have required too much effort for the three participants. On the other hand, the answers of the completed questionnaires were even more complete.

6.5. Threats to validity

In the following, the construct validity, internal validity and external validity of the evaluation is discussed. The construct validity of the evaluation is strengthened through the detailed explanation of the functionalities of the automation tool and the fine granular instructions of the usage scenarios. In the questionnaire, only questions were asked that related to the actions performed by the participant, to reduce the room for misunderstandings. Furthermore, the participants had the possibility to give a neutral answer to misleading questions and to write the reason for it in the free text field. Thus, ambiguities in the question were noticed and could be taken into account accordingly in the evaluation. An attempt was made to comply with the internal validity of the evaluation by always strictly adhering to the evaluation guideline. Particular attention was paid to ensuring that the automation tool always had the same initial status and was reset afterwards. During the explanation of the inputs and outputs of the tool, only the prewritten text was used and no other informations were given. The different response time of each participant regarding the questionnaire could be a confounding variable. For some participants it took several days to complete the questionnaire and for others the submission took place twenty minutes later. In retrospect, it should have been recorded which participant took how much time between the evaluation and the submission of the questionnaire, to understand whether the response quality or the response scope is related to it. Regarding external validity, just like with the requirements analysis, the small sample size and the fact that all participants were volunteers do not speak for a satisfactory external validity, because the general population may not be represented very well. Furthermore, some of the participants of the evaluation also took part in the requirements analysis, which means that so the significance of the results should be treated with caution.

7. Conclusion and outlook

This chapter summarizes the results of this work and gives an outlook that describes possible extensions of the automation tool and the implications of this work.

7.1. Conclusion

In this thesis, we aimed to collect requirements for an appropriate application infrastructure and to develop an automation tool that generates this application infrastructure, deploys the ML application and allows to observe the behavior of a new ML application version based on real-time user data.

To achieve this, we first conducted a comprehensive literature search by using a combination of termbased searches and snowballing. It turned out that the behavior of an ML application can be assessed against various monitoring objectives, such as business performance, application performance or model inputs and outputs. Each of the objectives can be measured by specific metrics, such as click-through-rate, inference time and prediction error rate. We found out that canary deployment, A/B testing or shadowing can be used to enable monitoring under the influence of real-time user data. Furthermore, we noticed that the monitoring practices thresholding and data slicing were frequently recommended in the literature.

Second, we carried out a target-performance analysis in the department of artificial intelligence at MHP, to find out how ML applications are currently operated productively and to what extent the behavior of a new ML application version is already being observed. We found out that the productive operation of an ML application requires a scalable application infrastructure, which can be implemented by using different AWS services that are configured and created by using infrastructure as code. We discovered that none of the monitoring methods that were identified in the literature search are currently used in the department. However, blue-green deployments have already been used in several projects as a means of quality assurance. Based on the results of the analysis, we elicited prioritized requirements for the infrastructure and the automation tool. We specified that the automation tool should configure and create a scalable application and training infrastructure, that it enables the execution of a canary and blue green deployment, and that it allows to view specific monitoring metrics.

Third, we implemented the automation tool according to the specified requirements. We developed a python console application that utilizes the AWS SDK to configure and create the required AWS services by using infrastructure as code. The tool is operated by first selecting the desired functionality and then entering the required inputs. Subsequently, the progress and the results of the automation tool are shown in the console output. The quality of the object-oriented implementation was assured through static code analysis, unit tests and integration tests.

Fourth, the evaluation of the automation tool is conducted by employees of the department according to the TAM model under the aspects of perceived ease-of-use, perceived usefulness

and behavioral intention. During the evaluation, the participants were first briefed about the functionality of the automation tool and then had to perform predefined usage scenarios independently. Finally, a questionnaire on the evaluating criteria had to be filled out. 81% of the responses to the evaluation were positive, which suggests that the automation tool performs its tasks satisfactorily. Although the evaluation was conducted online and screen control was shared with the interviewees, there were no serious technical complications. The use of google forms to distribute and fill out the questionnaires was also unproblematic.

7.2. Outlook

In the following, possible extensions of the automation tool are presented first and then the implications of this work are illustrated.

Possible extensions

[**Increased flexibility of settings**] The functionalities of the automation tool in its current form can only be adjusted by a small number of settings. For the use in different projects with different requirements it is necessary to increase the flexibility of the functionalities. Regarding the scalability, more metrics should be available as triggers and it should be possible to choose between vertical, horizontal and no scaling. Furthermore, the developer should be able to choose whether the scaling notification are shown on a dashboard or whether they are sent via email. With regard to the training infrastructure, the developer should be able to select whether the training data is versioned and whether the training results should be reported. Also, there should be an option to allow for periodic retraining.

[**Deployments with tests and rollbacks**] During a canary or blue-green deployment it would be essential to allow for customizable tests to be run on the new ML application version. Based on the results of the tests, the deployment should then be either completed or aborted. In case of a termination, already created resources should be removed and the user traffic should be redirected back to the current ML application version. Furthermore, it should be possible to make the deployment dependent on a critical metric. For example, if the error rate of the new ML application version is particularly high, the old ML application version should automatically be restored.

[**Usage of command line parameters**] For the integration of the automation tool into a CI/CD pipeline, the usage of command line parameters would be necessary. Currently the tool receives user input through sequential querying and subsequent receiving. For each developer input, a command line parameter would have to exist, which would then be appended to the python script in the console. This would make it possible to configure and execute a functionality using a single command.

[**Web interface**] The automation tool must currently be installed on a computer that runs python, docker and the AWS client tools. The AWS client tools furthermore have to be configured to allow access to AWS. The time required to install the automation tool is thus not negligible. It would be possible to run the automation tool itself in AWS and allow access via a web interface. This would eliminate the installation for the developer and would allow to start using the tool immediately.

[**Elimination of vendor lock-in**] Although the automation tool is currently tied to the

use of AWS, the binding should be loosened to support other cloud providers like Google Cloud Platform or Microsoft Azure. This would increase the number of potential users and the applicability to different projects. The current approach of infrastructure as code can be further pursued, but instead of CloudFormation a provider independent IaC tool like Terraform would have to be used.

Implications of this work

The approach for monitoring of new ML application versions under the influence of real time user data is not yet uniformly regulated within the department. The results of the interviews show that the methods of the literature search (canary deployment, A/B testing and shadowing) are only used sporadically among the projects or not at all. In order to provide guidance on existing procedures and their purpose, educational work must be provided. For this, a central document could be created that would represent the first port of call for employees. Another opportunity would be to organize workshops in which practical examples are performed and discussed.

To further increase the applicability of the above mentioned methods without having to rely on the development of individual software, a broadly used CI/CD deployment tool, such as Bamboo, would have to be complemented with an appropriate extension. This would make it possible to configure a canary deployment directly in the build plan of a Bamboo deployment job, without having to manually operate an additional tool. Furthermore, all end users of Bamboo would benefit from the feature and not just the MHP department. Even though Bamboo itself is not open source, by using its plugin functionality, such an extension could be realised.

Appendices

A. Literature overview

Authors	Abstract	Author keywords	Context and motivation	Research questions (RQs) and problems	Principal ideas and result	Contribution
Muthusamy, V. Slominski, A.	The stochastic nature of artificial intelligence (AI) models introduces risk to business applications that use AI models without careful consideration. This paper offers an approach to use AI techniques to gain insights on the usage of the AI models and control how they are deployed to a production application.	Artificial Intelligence, machine learning, microservices, business process	Context: Artificial intelligence (AI), including deep learning, have revolutionized business applications in diverse fields, including finance, manufacturing, and logistics. For businesses, adopting AI presents an opportunity and a risk. On the one hand, AI can reduce cost or provide better customer experience. On the other hand, adopting AI models present risks that can manifest in the form of monetary or reputation loss. Motivation: The authors want to reduce the risk of artificial intelligence models adversely affecting a business.	RQ: How to reduce the risk of artificial intelligence models adversely affecting the business? Problems: The effect of AI models on business applications can be unpredictable due to the stochastic nature of many machine learning models.	Principal idea: The authors develop an architecture that allows a set of AI algorithms to be transparently plugged into a business application. Result: An architecture that allows new models to be promoted carefully to production without breaking the application. It allows to track their performance over time with the assurance to get early warning about unexpected behaviour and the possiblity to quickly, and in many cases automatically, roll back to previous versions.	The article contributes an architecture that can be used to minimize the negative impact of AI models.
Breck, E. et Al.	Creating reliable, production-level machine learning systems brings on a host of concerns not found in small toy examples or even large offline research experiments. Testing and monitoring are key considerations for ensuring the production-readiness of an ML system, and for reducing technical debt of ML systems. But it can be difficult to formulate specific tests, given that the actual prediction behavior of any given model is difficult to specify a priori. In this paper, we present 28 specific tests and monitoring needs, drawn from experience with a wide range of production ML systems to help quantify these issues and present an easy to follow road-map to improve production readiness and pay down ML technical debt.	Machine Learning, Testing, Monitoring, Reliability, Best Practices, Technical Debt	Context: As machine learning (ML) systems continue to take on ever more central roles in real-world production settings, the issue of ML reliability has become increasingly critical. ML reliability involves a host of issues not found in small toy examples or even large offline experiments, which can lead to surprisingly large amounts of technical debt. Testing and monitoring are important strategies for improving reliability, reducing technical debt, and lowering long-term maintenance cost. Motivation: The authors want to quantify issues of production ML systems and present an easy to follow road-map to improve production readiness and pay down ML technical debt.	RQ: What should be tested regarding the use of ML in a production setting and how much is enough? Problems: ML system testing is a more complex challenge than testing manually coded systems, due to the fact that ML system behavior depends strongly on data and models cannot be strongly specified a priori.	Principal idea: The authors collect knowledge out of engineering decades of production-level ML systems at Google, in systems such as ad click prediction and the Sibyl ML platform. Result: A set of 28 actionable tests and a scoring system to measure how ready for production a given machine learning system is.	The article contributes a test set that acts as an easy to follow road-map to improve production readiness and pay down ML technical debt.
Zhang, Jie M. et al.	This paper provides a comprehensive survey of Machine Learning Testing (ML testing) research. It covers 144 papers on testing properties (e.g., correctness, robustness, and fairness), testing components (e.g., the data, learning program, and framework), testing workflow (e.g., test generation and test evaluation), and application scenarios (e.g., autonomous driving, machine translation). The paper also analyses trends concerning datasets, research trends, and research focus, concluding with research challenges and promising research directions in ML testing.	Machine learning, software testing, deep neural network	Context: The prevalent applications of machine learning arouse natural concerns about trustworthiness. Safety-critical applications such as self-driving systems and medical treatments increase the importance of behaviour relating to correctness, robustness, privacy, efficiency and fairness. With the recent rapid rise in interest and activity, testing has been demonstrated to be an effective way to expose problems and potentially facilitate to improve the trustworthiness of machine learning systems. Motivation: The paper seeks to provide a comprehensive survey of ML testing and software testing solutions for improving the trustworthiness of machine learning systems. The autors want to help software engineering and machine learning researchers to become familiar with the current status and open opportunities of and for of ML testing	RQs: What is the definition of machine learning testing? What is the current research state of machine learning testing? What are the challenges of ML testing? What are promising research directions? Problems: Machine learning are difficult to test because they are designed to provide an answer to a question for which no previous answer exists (Oracle Problem). The behaviours of interest for machine learning systems are also typified by emergent properties, the effects of which can only be fully understood by considering the machine learning system as a whole.	Principal idea: The authors define machine learning testing and they collect and analyze papers about the topic. They give an overview about challenges regarding ML testing and an outlook about ML testing in the future. Results: A definition of Machine Learning Testing, overviewing the concepts, testing workflow, testing properties and testing components. A comprehensive survey of 144 machine learning testing papers, across various publishing areas such as software engineering, artificial intelligence, systems and networking, and data mining. An analysis of the papers regarding their research distribution, datasets, and trends. An outlook that identifies challenges, open problems, and promising research directions for ML testing, with the aim of facilitating and stimulating further research.	The survey provides a comprehensive overview and analysis of research work on ML testing.

Table A.1.: Literature overview (1)

Authors	Abstract	Author keywords	Context and motivation	Research questions (RQs) and problems	Principal ideas and result	Contribution
Baylor, D. et al.	Creating and maintaining a platform for reliably producing and deploying machine learning models requires careful orchestration of many components - a learner for generating models based on training data, modules for analyzing and validating both data as well as models, and finally infrastructure for serving models in production. [...] Unfortunately, such orchestration is often done ad hoc using glue code and custom scripts developed by individual teams for specific use cases, leading to duplicated effort and fragile systems with high technical debt. We present TensorFlow Extended (TFX), a TensorFlow-based general-purpose machine learning platform implemented at Google. [...]	Large-scale machine learning; end-to-end platform; continuous training	Context: More and more organizations adopt machine learning as a tool to gain knowledge from data across a broad spectrum of use cases and products. Motivation: Having an appropriate machine learning platform enables teams to easily deploy machine learning in production for a wide range of products,that ensures best practices for different components of the platform, and limits the technical debt arising from one-off implementations that cannot be reused in different contexts.	RQ: How can machine learning be deployed in production? Problems: There are multiple problems. Products can have substantially different needs in terms of data representation, storage infrastructure, and machine learning tasks. Also, the platform has to support the case of training a single model over fixed data, but also the case of generating and serving up-to-date models through continuous training over evolving data	Principal idea: The paper presents the anatomy of end-to-end machine learning platforms and introduces TensorFlow Extended (TFX), one implementation of such a platform that the authors built at Google to address the aforementioned challenges. They describe the key platform components and the salient points behind their design and functionality. They also present a case study of deploying the platform in Google Play and discuss the lessons that they learned in this process. Results: TFX, a platform with TensorFlow-based learners and support for continuous training and serving with production-level reliability. Also, several best practices for using TFX.	The paper provides a machine learning platform to easily deploy machine learning applications in production, as well as best practices for it that are of general interest to researchers and practitioners in the field
Studer, S. et al	We propose a process model for the development of machine learning applications. It guides machine learning practitioners and project organizations from industry and academia with a checklist of tasks that spans the complete project lifecycle, ranging from the very first idea to the continuous maintenance of any machine learning application. With each task, we propose quality assurance methodology that is drawn from practical experience and scientific literature and that has proven to be general and stable enough to include them in best practices. We expand on CRISP-DM, a data mining process model that enjoys strong industry support but lacks to address machine learning specific tasks.	Machine Learning Applications; Quality Assurance Methodology; Process Model; Automotive Industry and Academia; Best Practices; Guidelines	Context: The authors created another paper about a process model for data mining, called "CRISP-DM" Motivation: The authors have identified two major shortcomings of CRISP-DM. First, CRISP-DM does not cover the application scenario where a ML model is maintained as an application. Second, CRISP-DM lacks guidance on quality assurance methodology. Now they want to expand on CRISP-DM to address machine learning specific tasks.	RQ: How can CRISP-DM be adapted to machine learning specific tasks? Problems: -	Principal idea: The authors follow the principles of CRISP-DM, to create a process model for the development of practical ML applications, that fulfills ML specific requirements and proposes a quality assurance methodology. Result: A process model that is called **CR**oss-**I**ndustry **S**tandard **P**rocess for the development of **M**achine **L**earning applications with **Q**uality assurance methodology (CRISP-ML(Q)).	A process model for machine learning applications, that helps organizations to increase efficiency and success rate in their machine learning projects.
Lwakatare, Lucy E. et al.	Artificial intelligence enabled systems have been an inevitable part of everyday life. However, efficient software engineering principles and processes need to be considered and extended when developing AI-enabled systems. The objective of this study is to identify and classify software engineering challenges that are faced by different companies when developing software-intensive systems that incorporate machine learning components. Using case study approach, we explored the development of machine learning systems from six different companies across various domains and identified main software engineering challenges. The challenges are mapped into a proposed taxonomy that depicts the evolution of use of ML components in software-intensive system in industrial settings. Our study provides insights to software engineering community and research to guide discussions and future research into applied machine learning.	Artificial intelligence, Machine learning, Software engineering, Challenges	Context: The application areas of ML to real-world problems are vast and range from large use in recommendation systems of social and e- commerce services, to highly regulated products, such as autonomous vehicle prototypes.The development of AI-enabled applications in real-world settings is non-trivial and the development process differs from that of traditional software. Motivation: There is a growing interest and need to understand how AI-enabled applications are developed, deployed and maintained over-time in real world commercial.	RQ: How to identify and classify challenges when developing software-intensive systems that incorporate machine learning components? Problems: -	Principal idea: The authors conducted an interpretive multiple-case study, to provide a deeper understanding of SE challenges for developing and operating ML systems in real-world commercial settings. Results: A description of the development process of six AI-enabled applications across various domains. A taxonomy to depict evolution in the use of ML components in commercial software-intensive systems. A classification of most important challenges at each stage of the evolution in the use of ML components in software-intensive systems.	The paper provides insights into the development and challenges of industrial ML systems.

Table A.2.: Literature overview (2)

Authors	Abstract	Author keywords	Context and motivation	Research questions (RQs) and problems	Principal ideas and result	Contribution
Atwal, H.	[...] DataOps methodology is the best way to eliminate barriers, collaborate, and maximize the chances of success. DataOps turns data science and analytics from the craft industry it is today in most organizations into a slick manufacturing operation. DataOps enables rapid data product development and creates an assembly line that converts raw data from multiple sources into production data products with a minimum of waste.[...]	Data science problems, data strategy, Lean thinking, Agile Collaboration, Build Feedback and Measurement, Building trust, DevOps, DataOps, Organizing, Technology, Factory	Context: DataOps is a collaborative data management practice focused on improving the communication, integration and automation of data flows between data managers and consumers across an organization. The goal of DataOps is to create predictable delivery and change management of data, data models and related artifacts. Motivation: Most organizations still approach data science as a series of large bespoke, waterfall research projects with artificial constraints in the provision of data when instead data-driven decisions can be automated, scalable, reproducible, testable, and fast.	RQ: How to adopt DataOps as a solution for delivering data science in an organization? Problems: Knowledge gaps, outdated approaches to managing data and producing analytics, and a lack of support for data analytics within the organization	Principal idea: The author describes current challenges with delivering data science, introduces Lean Thinking and agile methodology, explains how to build trust in data and offers recommendations to evaluate the technology to support DataOps objectives for agility and self-service. Result: A book with four main parts and ten chapters in total, that aims to challenge existing approaches of delivering data science and analytics by introducing a relatively new methodology that is much more flexible to adapt to future change.	With this book, data scientists can learn how to automate the testing and deployment of their data products and CIOs can measure the impact of their teams to adapt their business strategy.
Arnold, M. et al.	Today's AI deployments often require significant human involvement and skill in the operational stages of the model lifecycle, including pre-release testing, monitoring, problem diagnosis and model improvements. We present a set of enabling technologies that can be used to in- crease the level of automation in AI operations, thus lowering the human effort required. [...]	Automating AI, Operations Lifecycle, Performance Prediction, KPI Analytics, AI Operations, Pre-Deploy Test, Monitor, Diagnose and improve	Context: The end-to-end AI lifecycle consists of many often complex stages, including data preparation, modeling, and operations. While the details may vary from instance to instance, the overall flow often consists out of the same stages. Motivation: A lot of attention in both academia and industry has been focused on the earlier data science stages of the lifecycle.The final stages in AI operations are often neglected, or even overlooked entirely, despite being critical to the successful use of AI models in real-world applications.	RQ: How to increase the level of automation in the AI operations lifecycle? Problems: Pre-release tests are expensive to create and hard to keep updated. Manual labeling is costly. Monitoring is often done manually.	Principal idea: The authors develop technologies that capture aspects of production performance and they show how these technologies can be used to drive automation during operations. Result: A set of enabling technologies for (performance prediction and KPI analytics) that can be used to increase the level of automation in the four operation stages: pre-deploy test, deploy, monitoring and improvement.	The papers facilitates the automation of AI operations by providing different enabling technologies.
Bosch, J.	Deploying machine-, and in particular deep-learning, (ML/DL) solutions in industry-strength, production quality contexts proves to challenging. This requires a structured engineering approach to constructing and evolving systems that contain ML/DL components. In this paper, we provide a conceptualization of the typical evolution patterns that companies experience when employing ML/DL well as a framework for integrating ML/DL components in systems consisting of multiple types of components. In addition, we provide an overview of the engineering challenges surrounding AI/ML/DL solutions and, based on that, we provide a research agenda and overview of open items that need to be addressed by the research community at large.	AI systems, Conceptualizing AI Engineering, data science, cyber physical systems, safety-critical systems	Context: Over the last decade, the prominence of artificial intelligence (AI) and specifically machine- and deep-learning (ML/DL) solutions has grown exponentially. Motivation: Their research shows that the transition from prototype to industry-strength, production-quality deployment of ML/DL models proves to be challenging for many companies.	RQ: How can typical evolution patterns of ML/DL systems be conceptualized? What are engineering challenges surrounding ML/DL solutions? What is the current state of research in AI engineering? Problems: Few, if any, models exist that seek to create a structure and conceptualization of the problem space of AI development.	Principal ideas and results: First, the authors provide a conceptualization of the typical evolution patterns that companies experience as well as a framework for integrating ML/DL components in systems consisting of multiple types of components. Second, they provide an overview of the engineering challenges surrounding ML/DL solutions. Third, they provide a research agenda and overview of open items that need to be addressed by the research community at large.	The papers provides evolution patterns and engineering challenges of ML/DL systems and the current research state of AI engineering.
Sato, D. et al.	Machine learning applications are becoming popular in our industry, however the process for developing, deploying, and continuously improving them is more complex compared to more traditional software, such as a web service or a mobile application. They are subject to change in three axis: the code itself, the model, and the data. Their behaviour is often complex and hard to predict, and they are harder to test, harder to explain, and harder to improve. [...]	Automating AI, Operations Lifecycle, Performance Prediction, KPI Analytics, AI Operations, Pre-Deploy Test, Monitor, Diagnose and improve	Context: Continuous Delivery for Machine Learning (CD4ML) is a software engineering approach in which a cross-functional team produces machine learning applications based on code, data, and models in small and safe increments that can be reproduced and reliably released at any time, in short adaptation cycles. Motivation: The authors previously published a case study from a client project. They decided to build a sample ML application based on a public problem and dataset to illustrate a CD4ML implementation, as they were not allowed to use examples from real client code.	RQ: How to automate the end-to-end lifecycle of Machine Learning applications? Problems: different teams might own different parts of the process, and there is a hand over without clear expectations of how to cross these boundaries. Also, the process is hard to make reproducible and auditable, because of a variety of tools and large artefacts.	Principal idea: The authors describe the technical components they found important when implementing CD4ML, using a sample ML application to explain the concepts and demonstrate how different tools can be used together to implement the full end-to-end process. They also discuss further areas of development and research. Result: The technical components for CD4ML (Discoverable and Accessible Data, Reproducible Model Training, Model Serving, Testing/QA, Experiments Tracking, Model Deployment, CD Orchestration and Model monitoring) and example implementations.	The article provides help to automate the lifecycle of ML applications

Table A.3.: Literature overview (3)

B. Interview questionnaire

Section	Subsection	Is		Should	
		Questions and Answers		Questions and Answers	
INFRASTRUCTURE	DESIGN	7a	What does the infrastructure for the deployment of ML applications in the current project consist of?	7b	What does an optimal infrastructure for the deployment of an ML application look like? Why?
		Cloud provider: AWS is used as a cloud provider in five of eight projects. All AWS services used and reasons for using them in the projects: - Sagemaker: Training (B, F, H, I) - S3: Data storage, model memory (C, H, I, J) - ELB: Load distribution (B, I, J) - ECS, ECR or EKS: Deployment (C, I, J) - Route53: DNS entries (B, I, J) - EC2: Deployment (H, I) - CloudTrail: Logging (H) - Glue/Athena, CodeBuild: Data Processing (H) - EMR: Spark Cluster (E) - Lambda: Periodic data download (I) - SNS: Email dispatch (I) Tools used and reasons for use: - MetaFlow: Model training, structuring of Data Science workflows (C, H) - Terradata: Use as DBMS (A) - Cloudera: Use as platform for data engineering (A) Infrastructures: Project W: Training with Tensorflow and Docker on Sagemaker, deployment on EC2, load balancer and Route53 for DNS. Project X: Deployment of Sagemaker with Jupyter notebooks, S3 buckets for data storage, CloudTrails for logging, Glue/Athena for data processing, MetaFlow for model training, instead of combining Sagemaker and EC2 Project Z: Transfer of data from Google Bigquery to S3 via Lambda using Cronjob, then preprocessing on CodeBuild, then saving on S3 and subsequent training on Sagemaker. Models are stored on S3. Application deployed via ECS (individual software). ECR for containers. Spark Cluster on EMR. EC2 for execution. Application of Load Balancing and Route53. SNS for email dispatching. Project Y: Deployment on EKS in the form of microservice with REST endpoint. Model storage in S3. Use of database for storage of all model versions in S3. Load-Balancer for load distribution on different EKS nodes. Route53 for DNS.		Optimal features would be: - High performance (RAM/GPU) required (A, D) - AWS as a cloud provider, for fast infrastructure creation (G) - Wide choice of frameworks, programming languages and visualization for more flexibility (A) - Quick access to notebooks during model development is important (A) Optimal structure would be: - S3 Bucket for data storage and Kubernetes for preprocessing. Container for ML-Training. S3 as model storage. This would be a good basic infrastructure (C) - Possibility to use a pure EC2 instance for developers who prefer less abstraction (D) - Provisioning of different stages (Dev, Test, Pre-Prod, Prod) with sufficient capacities, as these are often not available (E)
	CREATION	8a	How is the infrastructure created in the current project?	8b	Would you find a tool helpful that automatically creates the infrastructure? If so, why? What should it look like?
		Of the four projects to which there was a response, three projects used Terraform and one used CloudFormation to create the infrastructure.		Of the ten interviewees, seven had an opinion and agreed that infrastructure creation using a tool contributes to effort reduction (A), time savings (B, D, G), reproducibility (B, C, F, I) and error prevention (F). Reference was always made to existing tools (e.g. CloudFormation and Terraform) and no new form of a tool was described.	
		9a	What problems are there when creating the infrastructure?	9b	What possible solutions can you imagine?
		Problems relating to infrastructure construction: - Automation is not possible without appropriate know-how (A) - Managing multiple accounts in AWS simultaneously is problematic. (H) - There is no reasonable verification of the CloudFormation templates without deployment (H) - There are role and authorization problems (I) - Complexity of the infrastructure (J)		Possible solutions: - Know-how for automation can be gained through communication with colleagues (A) - The management of several accounts simultaneously in AWS can be solved by developing individual software (H) - The solution to the role and authorization problems is a trial and error procedure (I) - In order to reduce the management effort of the infrastructure, services controlled by AWS can be increasingly used (J)	
		25a	What are the differences between the development of a prototype and an application that is put into production?		
		During the development of a prototype, the load on the infrastructure is significantly lower than in a productive application (F). This means that the hardware costs are also lower (F). Furthermore, the infrastructure does not have to be connected to the Internet, as there are no external accesses (F). With a productive application, more emphasis is also placed on monitoring and traceability (G). A structured approach to version control is also more important here (G). Scalability and security are particularly important in a production application (I).			
	NFR	10a	Which non-functional infrastructure requirements are specified in the current project?		
		Non-functional requirements: - Scalability (B, C, E, F, H, J) - Data protection (B, C, F, J) - Reaction time (B) - Modularity (E) - Model quality (F) - Multi-Tenancy (cross-account authorizations) (H)			

Table B.1.: Results for questions related to infrastructure

DEPLOYMENT	PROCEDURE	11a	How is the ML application deployed?	11b	Would you think a tool for automatic deployment is helpful? If so, why? What should it look like?
		Project V: Artifact is transferred via build server to application server and started by script. (A) Project W: There are three pipelines for pre-processing, model training and service delivery. They are fully automated and are triggered automatically when data changes, model quality deteriorates, at specific times, or commit-driven (B) Project X: Jupyter Notebook or pure Python code is deployed via MetaFlow (H). After the training the model is stored in S3 Bucket (C) Project Z: A commit on the master branch triggers an Oozie workflow in Bamboo to deploy the application (E) Project Q: There is a pipeline in Concurs and Cloud Foundry is used (F) Project S: Bamboo is used as a build server and triggered by commit. Gravity and CloudFoundry are used. (G) X, Z, Q and S use tools such as MetaFlow, Bamboo, Concourse, CloudFoundry and Gravity for deployment. W, Z and S trigger the deployment commit-driven.			All 8 interviewees who responded think that a tool is helpful. The reasons for this are time saving (A), automatic test execution (E, F, G) and automatic triggering (B). One interviewee finds automatic deployment useful, but prefers manual deployment to have more control (D). Suggestions for the form of the tool were not given, reference was made to existing tools such as Bamboo (A, E, G, H), CloudFoundry (A) and CodePipeline (H).	
		12a	What problems are there with the deployment of the ML application?	12b	What possible solutions can you imagine?
		Problems: - The configuration of the settings for test and production environments is often incorrect. For example, incorrect addresses or database constraints are configured (A) - Versioning data and ensuring model quality during deployment can be problematic (H) - No problems (B, C, E, F, S)			With regard to the versioning of data, there are third-party tools (E). Blue-green deployment could be used to ensure model quality (H).
		13a	What are the differences in deployment when developing a prototype and an application that is put into production?		
		Differences in the deployment of a prototype and a productive application: - No automatic data processing is used in a prototype and there is no contact with users (B) - A prototype may not be deployed at all. It is sufficient if certain evaluation metrics are determined on the own laptop to show the customer whether a use case is doable (C) - A prototype does not require a release and therefore there is no release process compared to a productive application. Zero-Downtime-Deployment is also not necessary (F)			
	QUALITY ASSURANCE	14a	Are methods used to check the quality of a model before going live?	14b	Should these methods be (still) used in the future? Why?
		Methods for quality control before going live: Project Y: Offline evaluation metrics Project W: During training, Precision, Recall and F1-Score are monitored Project X: There are functional tests to check if the predictions of the model are valid. Furthermore it is checked whether the model has been trained with the correct parameters (C). The test results of the new and old model are compared (D) Project Z: Yes, during training the properties of the model are monitored. There is an alarm functionality (E) Project Q: There is a dashboard with metrics on the quality of the predictions. This prevents a new model from being of lower quality than a previous model (F)			Five interviewees gave an answer and all think that quality review methods should be used in the future to detect changes in data and model quality (A), to maintain the quality characteristics (B, D, H) and to guarantee correct functionality (C).
		15a	If no quality assurance methods are used, why not?	15b	Under what circumstances are the methods feasible?
		To monitor the verification of the quality of the model during production operation, there must be users. In this case there were no users (D).			To monitor the quality of the model, quality criteria must be defined and test data must be available (A). Furthermore, a reasonable versioning of the model is required, which must be compatible with the deployment pipeline. A microservice architecture is also required in many cases where traffic (real-time user data) needs to be switched between different models (H).
		16a	If the interviewee does not know, then ask whether canary deployment, A/B testing or shadowing is used.		
		A/B Testing, Canary Deployment and Shadowing are not used in any current project. In previous projects A/B testing was performed by C. According to A, V, D and F, canary deployment is considered to be useful. F and H have already performed a Blue-Green deployment. The lack of A/B testing or canary deployment could be related to the fact that these methods may not sufficiently known or technically difficult to implement.			
		17a	Which errors can be prevented with a canary deployment, A/B testing or shadowing?		
		By using methods for quality assurance of the model before going live, changes in the data and model quality can be detected (A). In addition, software problems are detected (B) and server errors occur due to incompatibility of software and server (D). A/B testing can be used to assess changes in user experience (C). A blue-green deployment can also enable high availability (C).			

Table B.2.: Results for questions related to deployment

MONITORING	PROCEDURE	18a	Are properties of the ML application or ML model monitored?	18b	Should characteristics of the ML application or model be monitored in the future? If so, why?
		Project W: Yes Project X: Yes Project Z: Yes Project Q: Yes		Nine interviewees responded that they think that properties of the ML application or model should be monitored. This will allow changes in user behaviour, data distribution and model quality to be identified (A). If several models are used simultaneously, it is also important to monitor the query rate of the models (B). Scaling can also be simplified by monitoring (D). Furthermore, calls of the ML application for traceability and time measurement should be monitored. (G). The error rate of the model and all important fixed KPIs should be monitored (I), CPU and RAM consumption are also interesting for monitoring (E). In general, online monitoring is more meaningful than monitoring during training (H). Monitoring is helpful for quality assurance (C, J).		
		19a	How is the monitoring of properties from a technical point of view carried out in the majority of projects?	19b	How could the procedure be made even easier?
		Procedures for monitoring: - The evaluation metrics of a new model are compared with the old model after each training. Thresholding is used and alarms are triggered (C), - The CPU load is measured manually in Python - Results of the ML application are written in tables and then displayed via frontend in the form of a dashboard (E). - An ELK stack (Elastic Search, Log Stack and Kibana) is used within AWS (F). - Splank and Kibana are used in combination. The application either writes logs to files if Kibana runs on the same server. Alternatively Kafka is used as a message broker to send log messages to the server running Kibana. The log data can then be queried via Query Language. Often call IDs, user name, component, class and message are logged (G). - All requests can be logged and displayed on a dashboard. Alternatively, certain metrics can be displayed per model (H). - CloudWatch can be used for logging and alerting. Together with integration via a slack channel, the alert messages reach the recipients (I).		The use of cloud services could facilitate monitoring (D), an example is the load monitoring in AWS (J). The use of thresholding and alerting during monitoring is seen as useful (I).		
		20a	If properties are monitored, what problems are there with monitoring in the majority of projects?	20b	What possible solutions can you imagine?
		A universal evaluation metric such as Accuracy is not always useful or applicable (C). In monitoring, there are access problems (E) and the customer's specifications mean that you are bound by fixed tools (F). The accuracy of monitoring is also important to avoid fingerpointing. If a service has a poor performance, it could be because it internally accesses another service, which is solely responsible for the performance. So it must be monitored fine enough (F).		The fineness of monitoring a user call can be achieved by including the complete call stack (F).	
	METRICS	21a	Which characteristics of the ML application and the ML model are monitored?	21b	Which other properties should ideally also be monitored and why?
		Project W: Precision, Recall and F1-Score during training (B) Project X: Response time, accuracy (H) Project Z: Learning rate in training. It is checked whether current data is used. (E) Project Q: The model monitors metrics on the quality of the predictions (F).		It would make sense to monitor user behaviour, data distribution, model quality and all KPIs (A). Precision, Recall, F1-Score are useful for classification problems (C, D), Accuracy for numerical problems (C). The maximum and average of the data is useful to monitor (D). The frequency of the model calls and the distribution of the output values is useful (B). For scalability, monitoring the CPU load is useful (D).	
				22b	Which errors should be prevented by monitoring?
				User satisfaction should be given (A) and the probability distribution of expenditure should be satisfactory (B). Unrealistic inputs should be avoided (D) and the timeliness of the data should be ensured (E). Monitoring is helpful for error analysis, as errors can be better reproduced (H).	
	PRESENTATION	23a	Where are the results of the metrics presented?	23b	How should the metrics be presented optimally and why?
		A dashboard is used for all monitoring projects (B, D, E, F, G, H, I)		A dashboard is good (A, D, E, F, G, H, I) It should be clarified with stakeholders which metrics are presented (A). A clean definition of metrics is important (H).	
OVERALL				24b	Is there a core functionality among all projects that can be automated for infrastructure, deployment and monitoring? What is the core functionality?
				- Automation of data integration via pipelines for reproducibility would be helpful. The feedback loop from monitoring to model building is important. In general, deployment, infrastructure, and testing can be well simplified by automation (A). - An S3 bucket in combination with ECS would be a good basis (C). - No (D). - The different test stages should be pre-built and have guaranteed resources. It should be obvious why a build takes a long time and how long it takes to start (E). - A knowledge base for example architectures is useful and is currently being created (F). - Probably not. The customers have specifications. In sub-projects it might be possible (e.g. VPC creation in AWS), but generally there are few POCs on the own AWS infrastructure (G). - Data access, model development, deployment and production can be automated (H). - Monitoring for AWS could possibly be generalized (I).	

Table B.3.: Results for questions related to monitoring and the overall question

C. Class diagram of the automation tool

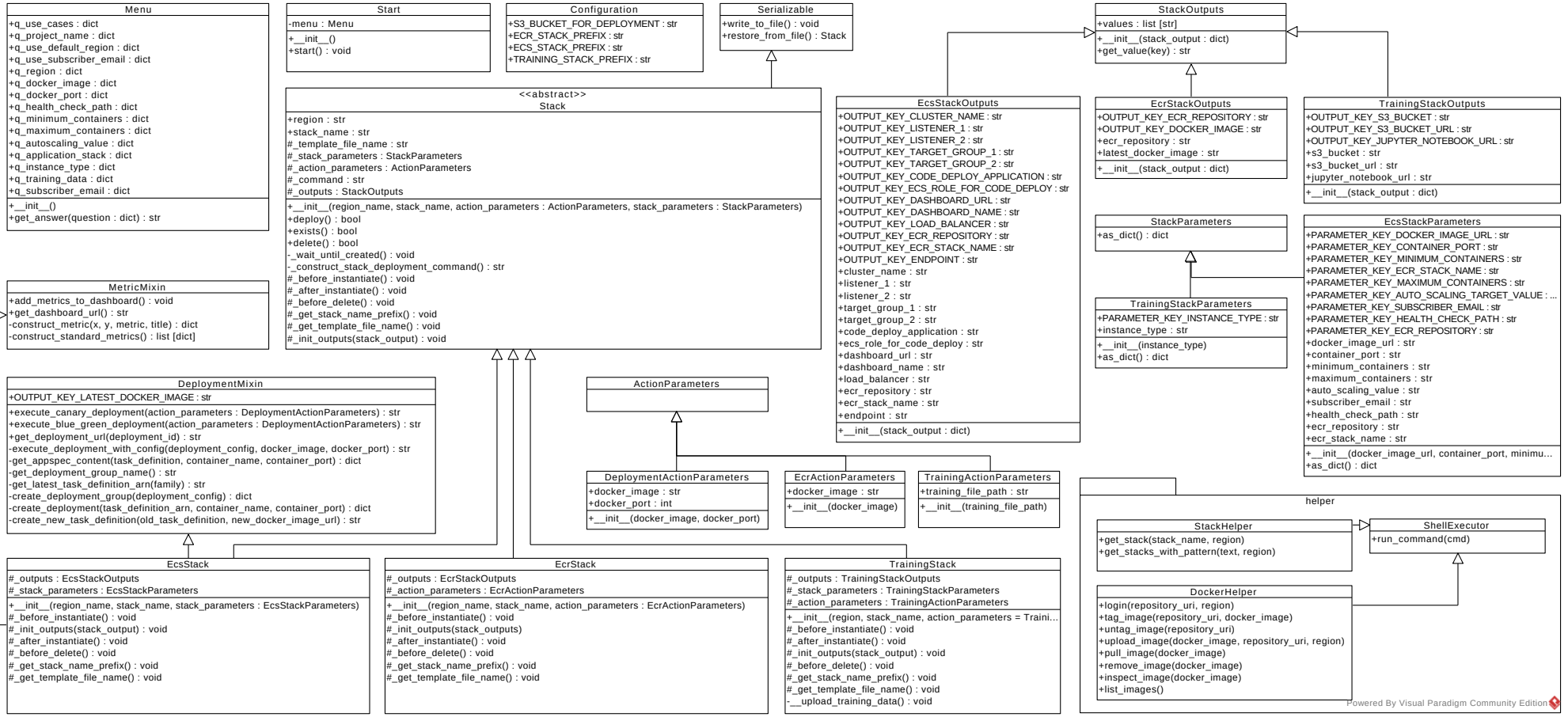


Figure C.1.: Class diagram of the automation tool

Class	Description
Menu	Contains all menu messages
Start	Entry point of the application. The core functionalities are triggered from here.
Configuration	Configures S3 deployment bucket name and stack prefixes
Serializable	Allows to save an object to a file and read it back
Stack	The central class representing the CloudFormation Stack
DeploymentMixin	Extends the functionality of the stack to be able to execute blue green and canary deployments
MetricMixin	Extends the functionality of the stack to view metrics on a dashboard
EcrStack	Represents first part of the application infrastructure. It creates an ECR repository with the docker image
EcsStack	Represents second part of the application infrastructure. Responsible for all remaining resources
TrainingStack	Represents the training infrastructure
StackParameters	Represents the stack parameter of a stack
EcsStackParameters	The stack parameters of the EcsStack
TrainingStackParameters	The stack parameters of the TrainingStack
ActionParameters	Represents the action parameters of a stack
EcrActionParameters	The action parameters of the EcrStack
TrainingActionParameters	The action parameters of the TrainingStack
DeploymentActionParamters	The action parameters for the DeploymentMixin
StackOutputs	Represents the outputs of a stack
EcsStackOutputs	Represents the outputs of the EcsStack
EcrStackOutputs	Represents the outputs of the EcrStack
TrainingStackOutputs	Represents the outputs of the TrainingStack
ShellExecutor	Executes commands in the console
StackHelper	Utility functions, such as creating a Stack object from a remote CloudFormation stack
DockerHelper	Docker utility functions, such as tagging an image

Table C.1.: Description of all classes of the automation tool.

D. Evaluation guideline

Section	Subsection	Evaluation Guideline	
INTRO		Thanks for the participation	
		Explanation of the procedure	
		Assurance of anonymity and data protection	
		Question whether session can be recorded	
BRIEFING		The automation tool functionality is briefly described and its inputs and outputs are explained	
USAGE SCENARIOS	CREATE AN APPLICATION INFRASTRUCTURE (*S1*)	In this scenario you create an application infrastructure for a given sample application. Start the console application with "python start.py.	
		RI1	Configure the application infrastructure like that: 1. Please select "Create an application infrastructure" first. 2. Confirm the default region with "y" 3. Enter a name for the project (e.g. "test-<your first name>"). 4. Select the docker image "example-image". 5. Confirm the proposal for port 80 6. Confirm the suggestion for "/" as health check path 7. Confirm the proposal for the minimum of one application instance 8. Enter "2" for the maximum number of application instances 9. For the CPU limit in percent please enter "1" instead of "80 10. Press "y" to be informed about scaling events via email 11. Enter "leon.radeck@web.de" as email The configuration of the application infrastructure is now completed.
		RI2	After entering the parameters, the creation of the application infrastructure should begin. It now takes about five minutes until the following line is displayed in the console: *Successfully created/updated stack - ml-app-test-<your first name> in eu-central-1* Check the console output if all resources were successfully created. The creating of the application infrastructure is now completed.
		RI5	You can view the current application instances by opening up the *TaskOverview* url in the console. Please check that there is currently one instance. Please execute the following command in four separate command lines, to trigger the scaling of the application infrastructure: *curl <endpoint>?[1-5000]* Please refresh the *TaskOverview* url in the browser until you see two application instances. This means that the scaling of the application infrastructure was successful.
		RM2	Please check that you received an email that notifies you about the scaling. Open up the email application and look under Web >> Unknown.
	CREATE A TRAINING INFRASTRUCTURE (*S2*)	In this scenario, you create a Sagemaker Notebook instance on which you run a Jupyter notebook, which uses your supplied training data to train a model and stores it in S3.	
		RI3	Configure the training infrastructure like that: 1. Start the console application with "python start.py" 2. Please select "Create a training infrastructure" first 3. Confirm the default region with "y" 4. Enter a name for the project (e.g. "test-<your first name>2"). 5. Enter "/Users/leonradeck/Documents/Master thesis/Evaluation/Scenarios/Scenario2/bank_clean.csv" as path for your training data. 6. Choose "ml.t2.medium" as a notebook instance type to vertically scale the notebook instance to 2vCPus and 4GB RAM The configuration of the training infrastructure is now completed.
		RI4	After entering the parameters, the creation of the application infrastructure should begin. It now takes about five minutes until the following line is displayed in the console: *Successfully created/updated stack - ml-training-test-<your first name> in eu-central-1* Check the console output if all resources were successfully created. The creation of the training infrastructure is now completed. To test whether an ML model can be trained, follow these steps: Please open up the link that is displayed in the console. Check that you get forwarded to a jupyter notebook environment. Follow these steps to check that you can train a model: 1. Click on "New" >> "conda_python3". 2. Under the path: ".../Evaluation/Scenarios/Scenario2/notebook.txt" you will find the code for the Jupyter notebook. Please copy the code from the file into the notebook. 3. Before you execute the code via "Run" and thus start the training of the model, you have to initialize the variables "bucket_name" and "training_file_name" with the correct values.

Table D.1.: Evaluation guideline (1)

			4. For "bucket_name" you enter the output value of the console at "S3Bucket" (e.g. "ml-training-test-leon"). 5. For "training_file_name" you enter "bank_clean.csv". 6. Now click on "Run". After some time the message *Starting - Starting the training job...* should be displayed. After 5 minutes the message *Training job completed* should appear. Please check now if the completed trained model (model.tar.gz) was uploaded to its S3 bucket in the folder /sagemaker/DEMO-xgboost-dm/output/. For easy access to the S3 Bucket, you can click on the S3 bucket link in the console output. If the model exists, this means that the training was successful.
		RI5	Copy the *NotebookInstanceURL* from the console output into the browser and press enter. Check that the notebook instance type is "ml.t2.medium". This means that the correct notebook instance was used during the training and your desired CPU and RAM specifications were met.
	EXECUTE A BLUE GREEN DEPLOYMENT (*S3*)	The application provided in *S1* is now to be replaced by a new application version. A blue-green deployment will be executed for this purpose.	
		RD3	Configure and execute the blue green deployment like that: 1. Open a new browser tab with the endpoint of the application from *S1* and click on "Auto Refresh". You can search for the endpoint in the outputs of the corresponding CloudFormation stack. The stack name follows the pattern "ml-app-test-leon". 2. Start the console application with "python start.py 3. Select "Execute a blue-green deployment". 4. Confirm the default region with "y 5. Select the created stack in *S1* 6. Select "example-image2" as the docker image. 7. Confirm port 80. 8. Confirm "/" as Health Check path Wait until a link appears in the console output and copy the link to the browser. You will be redirected to a page with deployment information. In the diagram on the right side you can see which of the two ML application versions is currently receiving user traffic. Wait until the replacement task traffic is "100%" and then check if the endpoint shows a green image instead of a blue one. You successfully executed a blue green deployment by shifting the complete user traffic to the new application version. To check the quality of the new application version, copy the DashboardURL of the console output into the browser and open it. Now check if the RequestError metric in the dashboard shows the number of requests of the application version and their error count by refreshing the endpoint url of the application a few times while looking at the metric. The metric should show no errors and the correct number of requests. Click on the following link [...] to generate an error. Now check if the metrics has changed and shows one error.
		RD1	Because the endpoint now shows a green image, this means that the new application version is running without problems and it is available to user via HTTP.
	VIEW MONITORING METRICS (S4)	In this scenario you look at metrics for an existing application infrastructure with an ML application containing an ML model. Follow these steps: 1. Start the console application with "python start.py 2. Select "View resource utilization metrics 3. Confirm the default region with "y 4. Select "ml-app-radioreco" 5. Click on the link that is output. 6. Select "1h" as the period in the upper right corner.	
		RM3	Check whether you can see metrics such as CPU/RAM usage of the application infrastructure.
		RM1	Check whether you can see the error rate and age of the ML application and ML model. The CPU/RAM usage of the application infrastructure corresponds to the CPU/RAM usage of the ML application and its model.

QUESTIONNAIRE	EQ	Req.	Variable	Nr.	Questions
	EQ1	RI1	Perceived easy-of-use	1	It is easy to make the settings for the application infrastructure.
			Perceived usefulness	2	The settings for the application infrastructure make sense.
			Behavioral intention	3	I will continue to make the settings for the application infrastructure this way in the future.
		RI2	Perceived easy-of-use	4	Creating the application infrastructure is easy.
			Perceived usefulness	5	The tool completely covers the task of creating the application infrastructure.
			Behavioral intention	6	I will continue to create application infrastructures in this way in the future.
		RI5	Perceived easy-of-use	7	The scaling of the application infrastructure is understandable and comprehensible.
			Perceived usefulness	8	The task of scaling the application infrastructure is fully covered.
			Behavioral intention	9	I will continue to use scaling for application infrastructures in the future.

Table D.2.: Evaluation guideline (2)

	EQ2	RI3	Perceived easy-of-use	10	It is easy to make the settings for the training infrastructure.
			Perceived usefulness	11	The settings for the training infrastructure make sense.
			Behavioral intention	12	I will continue to make the settings for the training infrastructure this way in the future.
		RI4	Perceived easy-of-use	13	Creating the training infrastructure is easy.
			Perceived usefulness	14	The tool completely covers the task of creating the training infrastructure.
			Behavioral intention	15	I will continue to create training infrastructures in this way in the future.
		RI5	Perceived easy-of-use	16	The scaling of the application infrastructure is understandable and comprehensible.
			Perceived usefulness	17	The task of scaling the application infrastructure is fully covered.
			Behavioral intention	18	I will continue to use scaling for training infrastructures in the future.
	EQ3	RD1	Perceived easy-of-use	19	The time of availability of the application via HTTP is easily visible for the users.
			Perceived usefulness	20	The task of making the application available to users via HTTP is fully covered.
			Behavioral intention	21	I will continue to make applications available to users via HTTP in this way.
		RD3	Perceived easy-of-use	22	The execution of a blue green deployment is simple.
			Perceived usefulness	23	The task of executing a blue green deployment is completely covered by the tool.
			Behavioral intention	24	I will continue to execute blue green deployments in this way in the future.
			Perceived easy-of-use	25	Checking the quality of the new application version is easy.
			Perceived usefulness	26	The task of checking the quality of the new application version is fully covered by the tool.
			Behavioral intention	27	I will continue to check the quality of the new application version this way.
	EQ4	RM1	Perceived easy-of-use	28	The metrics about the ML application are presented in a clear and understandable way.
			Perceived usefulness	29	The metrics about the ML application are useful to better understand the ML application.
			Behavioral intention	30	I will continue to view metrics about the ML application.
			Perceived easy-of-use	31	The metrics about the ML model are presented in a clear and understandable way.
			Perceived usefulness	32	The metrics about the ML model are useful to better understand the ML model.
			Behavioral intention	33	I will continue to view metrics about the ML model.
		RM3	Perceived easy-of-use	34	The metrics about the application infrastructure are presented in a clear and understandable way.
			Perceived usefulness	35	The metrics about the application infrastructure are useful to better understand the application infrastructure.
			Behavioral intention	36	I will continue to view metrics about the application infrastructure.
		RM2	Perceived easy-of-use	37	The automatic notification when the application infrastructure scales reaches me via the desired communication channel and is written in an understandable way.
			Perceived usefulness	38	The task of automatic notification when the application infrastructure scales is fully covered.
			Behavioral intention	39	I will continue to rely on notifications when the application infrastructure scales.

Table D.3.: Evaluation guideline (3)

E. Evaluation results

EQ	Evaluation Questionnaire Results (1) – Application Infrastructure	
EQ1	1	It is easy to make the settings for the application infrastructure.
	Strongly agree	Self-explanatory (A)
	Strongly agree	Clear, simple information required (G)
	Strongly agree	The tool is clearly understandable and very well structured! (E)
	Strongly agree	The steps in the console are easy to understand and have good default values. (D)
	Agree	Relatively intuitive, some little things (no confirmation with Enter after y/n, click on menu item leads to exception etc.) should possibly be changed. (C)
	Agree	Simple in principle. The workflow was easy to follow and the options were understandable. However, a certain amount of prior knowledge is required: - Set environment variables correctly or install library in environment - Using the Console - AWS Options But with the appropriate documentation, it should be easy to use. (F)
	Agree	Some applications need more specific settings and unique solutions (B)
	2	The settings for the application infrastructure make sense.
	Strongly agree	Easy to understand and clear (A)
	Strongly agree	The settings cover most scenarios well. (D)
	Agree	Like previous comment (B)
	Agree	For smaller applications the most important points are adjustable (C)
	Agree	Depending on the application purpose, not only CPU usage but also number of requests or RAM usage could be accepted as a trigger. Within the scope of your work the CPU usage is of course completely sufficient! (E)
	Agree	Every project has different requirements. A general setting for all projects is rather difficult. (F)
	Neutral	Yesterday I was wondering what happens if there are errors or wrong entries - there is probably more you can do in terms of user control. (G)
	3	I will continue to make the settings for the application infrastructure this way in the future.
	Strongly agree	(A)
	Strongly agree	(B)
	Agree	Yes (D)
	Agree	That depends strongly on the respective requirements. Of course, this can also differ and must be chosen according to the context. (E)
	Agree	The creation or configuration of the infrastructure is simple, quick to implement and, above all, repeatable! (F)
	Agree	It may well be that we will introduce something like this in the project (G)
	Neutral	Similar but more individually adjustable depending on the application. (C)
	4	Creating the application infrastructure is easy.
	Strongly agree	Self-explanatory (A)
	Strongly agree	(B)
	Strongly agree	Super easy! Click "Enter" a few times. Manually in AWS this would take forever. (E)
	Strongly agree	Everything automated - top! (G)
	Agree	Creating the application infrastructure was easy. (C)
	Agree	Just after creating the application infrastructure it was a bit confusing due to many generated URLs. If necessary this can be arranged better. (Table?) (D)
	Agree	The workflow makes creation easier and traceable. (F)
	5	The tool completely covers the task of creating the application infrastructure.
	Strongly agree	(A)
	Strongly agree	In any case! (D)
	Strongly agree	Absolute agreement ;-) (E)
	Agree	(B)
	Agree	I like the creation today - I would still be interested in the topic CF Stack Updates. So what happens if you create a new version of the stack in the same account and make updates with it. (G)
	Neutral	Here it depends on the requirements. From my point of view, for example, the following are missing:

Table E.1.: Results of questions regarding *EQ1*

	- Data integration (Which possibility do I have to address different data pools?) - Multi-User usage (Can other developers work with me?) - Is there a versioning? Security, e.g. access rights? (F)
Disagree	Not all details are individually adjustable for the creation of application infrastructure (C)
6	I will continue to create application infrastructures in this way in the future.
Strongly agree	Quick and easy (A)
Strongly agree	(B)
Strongly agree	The infrastructure is created in a uniform way. Errors are thereby avoided. (E)
Agree	Yes (D)
Agree	It is a cool tool and it can update / create whole systems with appropriate development (G)
Neutral	Up to now it was not necessary to set up an infrastructure, because this is implemented by other colleagues. However, if I had to do it, such an automation tool would be useful. (F)
Disagree	For many larger projects individual settings are necessary (C)
7	The scaling of the application infrastructure is understandable and comprehensible.
Strongly agree	(A)
Strongly agree	(B)
Strongly agree	There is nothing to add here. This basic knowledge should be assumed in any case. (D)
Strongly agree	Very understandable. The trigger for scaling is also more than clear! (E)
Strongly agree	Was understandable! (F)
Agree	The procedure with step-by-step instructions and checking the results makes the individual steps easy to follow (C)
Agree	Horizontal scalability cannot be used in all cases in our systems (G)
8	The task of scaling the application infrastructure is fully covered.
Strongly agree	With CloudWatch and emails very clear (A)
Strongly agree	(B)
Strongly agree	Yes, when the CPU load is reached, an additional instance is started. (E)
Agree	Typical scenarios are covered. (D)
Agree	Not only the main memory plays a role, but also hard disk space, system failures, etc. AWS offers these features by default. (F)
Agree	Good for services; however, horizontal scaling cannot be used in all system parts (G)
Neutral	Lack of knowledge whether all horizontal scaling settings can be adjusted (C)
9	I will continue to use scaling for application infrastructures in the future.
Strongly agree	Very comfortable (A)
Strongly agree	(B)
Strongly agree	Not only in the future, but also in the past always done ;-) (E)
Strongly agree	From a purely ecological point of view, scaling out or scaling in for peaks makes sense. The system becomes more robust through distribution/replication. Also, misjudgments can be intercepted. (F)
Agree	Always useful for applications with fluctuating access rates (C)
Agree	Yes, I don't know of any reason to the contrary. (D)
Neutral	Depends on the system and its requirements - cool would be a deployment as container in a Kubernetes or similar to get more properties directly out-of-the-box (G)
=>	General feedback about the application infrastructure
As mentioned above, the results of a creation are a bit confusing, maybe you can visualize the data in a more compressed way so that you have to scroll less. The handling was very intuitive like you are used to from other CLI. (D)	
I think the tool is really great! Depending on your requirements, it can take a lot of the work out of your hands and you can see very clearly that especially the infrastructure setup in the cloud can be wonderfully automated! (E)	
A really cool and useful tool. Since the requirements are constantly changing, the tool should definitely be flexible. As mentioned above, there are still some features missing, which are essential. (F)	
Cool implementation! good job :) (G)	

Table E.2.: Results of questions regarding *EQ1* (2)

EQ	Evaluation Questionnaire Results (2) – Training Infrastructure	
EQ2	10	It is easy to make the settings for the training infrastructure.
	Strongly agree	Self-explanatory (A)
	Strongly agree	(B)
	Strongly agree	(D)
	Strongly agree	The tool gives detailed instructions (E)
	Agree	As with application infrastructure well step-by-step guide to setting (C)
	Agree	See reason for application structure (F)
	Agree	I am no friend of Sagemaker, so I would rather deploy standard Python with Metaflow or similar but basically good idea. (G)
	11	The settings for the training infrastructure make sense.
	Strongly agree	Simple and sufficient (A)
	Agree	(B)
	Agree	Well explained but a data scientist will probably have his problems without support. (G)
	Agree	Yes, they cover most scenarios. (D)
	Agree	Same as before (E)
	Neutral	As far as the evaluation showed, not arbitrary training structures (e.g. training jobs with own maintainers etc.) can be set, for the use case it was sufficient (C)
	Neutral	See reason for application structure (F)
	12	I will continue to make the settings for the training infrastructure this way in the future.
	Strongly agree	Quick and easy (A)
	Agree	Yes, for sure. (D)
	Agree	Not my field of application. Therefore I will probably not create any training infrastructures in the future. (E)
	Agree	It is a good start. But there should be much more properties configurable for a project deployment I think. (G)
	Neutral	It is highly dependend on the type of data, algorithms, preprocessing ... (B)
	Neutral	I'm still missing the possibility to enter the resources (Where does the data come from?). (F)
	Disagree	In the current project IaC is available for the creation of training infrastructure (C)
	13	Creating the training infrastructure is easy.
	Strongly agree	See above (A)
	Strongly agree	(B)
	Strongly agree	Yes, I can only agree with that. (D)
	Strongly agree	Same as before (E)
	Agree	It is easy to create the training infrastructure. (C)
	Agree	See reason for application structure (F)
	Agree	It is easy and a first step towards complete automation - maybe add a Lamba-Function or regular "retraining" via AWS Step-functions (G)
	14	The tool completely covers the task of creating the training infrastructure.
	Strongly agree	(A)
	Strongly agree	Yes (D)
	Strongly agree	Absolute agreement here again (E)
	Agree	A basic training with clean data was provided (B)
	Neutral	Can I use any library that exists? Versioning available? Is there a possibility for reporting on the findings in the training? (F)
	Neutral	See comment above - Partial automation only the beginning of training infrastructures (G)
	Disagree	See above (C)
	15	I will continue to create training infrastructures in this way in the future.
	Strongly agree	Quick and easy (A)
	Agree	Yes (D)
	Agree	Quickly setting up a Jupyter notebook in the cloud and getting started is great! (F)

Table E.3.: Results of questions regarding *EQ2*

	Agree	I like the type of deployment (G)
	Disagree	It is highly depends on the type of data, algorithms, preprocessing ... (B)
	Disagree	See above (C)
	Disagree	Not my field of application. Therefore I will probably not create any training infrastructures in the future. (E)
	16	The scaling of the training infrastructure is understandable and comprehensible.
	Strongly agree	(A)
	Strongly agree	Is understandable and comprehensible. (F)
	Agree	(B)
	Agree	Yes (D)
	Agree	I like to use the mechanisms of Sagemaker, but it is not understandable what Sagemaker does with intermediate results, etc. (G)
	Neutral	The vertical scaling was not discussed in detail (C)
	Neutral	The vertical scaling went completely past me. I must have "clicked through" too fast. (E)
	17	The task of scaling the training infrastructure is fully covered.
	Strongly agree	(A)
	Strongly agree	Was understandable (F)
	Agree	Yes, for my requirements. (D)
	Agree	Same reason as above - traceability in one training run and several experiments is currently not good with Sagemaker. (G)
	Neutral	training on multiple instances? (B)
	Neutral	The vertical scaling was not discussed in detail
	Neutral	The vertical scaling went completely past me. I must have "clicked through" too fast. (E)
	18	I will continue to use scaling for training infrastructures in the future.
	Strongly agree	Quick and easy (A)
	Strongly agree	Scaling (vertically and possibly horizontally) brings speed advantages over non-scaled trainings (C)
	Agree	(B)
	Agree	Use yes - but rather via Metaflow or similar (G)
	Neutral	Depending on the respective scenario yes. (D)
	Neutral	Usually scaling is not necessary for exploration/training. If you have to go there every day for intensive training to have a model on the same day, both vertical and horizontal scaling is useful. (F)
	Disagree	I prefer horizontal scaling if possible (E)
	=>	General feedback about the training infrastructure
	Your tool is also very useful here. (E)	
	In itself really good, things like versioning of the code, user rights, optional scaling are missing? Or did I overlook that? (F)	

Table E.4.: Results of questions regarding *EQ2* (2)

EQ	Evaluation Questionnaire Results (3) – Blue green deployment	
EQ3	19	The time of availability of the application via HTTP is easily visible for the users.
	Strongly agree	(A)
	Strongly agree	(B)
	Strongly agree	Availability is indicated in the output (C)
	Strongly agree	Yes (D)
	Strongly agree	Clear display by the AWS GUI (E)
	Strongly agree	Visual dashboards are always good ;-) (F)
	Strongly agree	I like it (G)
	20	The task of making the application available to users via HTTP is fully covered.
	Strongly agree	(A)
	Strongly agree	(B)
	Strongly agree	Full agreement! (D)
	Strongly agree	Deployment of the application using the tool works properly (E)
	Strongly agree	It was always clear which application was running and when the new one would be made available. (F)
	Strongly agree	if you build it into a CI/CD chain, yes :) (G)
	Neutral	Lack of knowledge in the completeness of HTTP applications (security, performance, ...) (C)
	21	I will continue to make applications available to users via HTTP in this way.
	Strongly agree	Quick and easy (A)
	Strongly agree	The Blue-Green Deployment process alone is one of the best practices. The tool can implement exactly that. (F)
	Agree	Yes, definitely! (D)
	Agree	with some adaptation to Code-Deploy or Bamboo builds (G)
	Neutral	It depends on the application (B)
	Neutral	Lack of experience (C)
	Neutral	Depends as always on the requirement (E)
	22	The execution of a blue green deployment is simple.
	Strongly agree	Self-explanatory (A)
	Strongly agree	Understandable process (C)
	Strongly agree	Yes (D)
	Strongly agree	Absolutely simple (E)
	Strongly agree	Simple operation. (F)
	Strongly agree	Correct (G)
	Agree	(B)
	23	The task of executing a blue green deployment is completely covered by the tool.
	Strongly agree	(A)
	Strongly agree	(B)
	Strongly agree	Yes (D)
	Strongly agree	Fully functional (E)
	Agree	Blue green deployment worked (whether complete cannot be judged due to lack of experience)
	Agree	Unfortunately, I could not see the error. But it looked quite good. (F)
	Agree	Missing tests and abort scenarios for faulty tests (G)
	24	I will continue to execute blue green deployments in this way in the future.
	Strongly agree	Quick and easy (A)
	Strongly agree	Yes, because of the ease of use it makes sense. (D)
	Agree	(B)
	Agree	Sometimes you have not only one but several services. Here it would be nice to have an overview and control of all deployments of the services. (F)

Table E.5.: Results of questions regarding *EQ3*

	Agree	It is nevertheless a conceivable tool (G)
	Neutral	Lack of experience (C)
	Disagree	A Blue-Green deployment in projects is covered by CI/CD. Here, for example, a part of your tool could be used. (E)
	25	Checking the quality of the new application version is easy.
	Strongly agree	(A)
	Strongly agree	Yes (D)
	Strongly agree	Nothing to say (F)
	Agree	(B)
	Agree	Fits with the existing metrics (G)
	Neutral	Lack of experience (C)
	Neutral	I am not sure what you mean by quality? (E)
	26	The task of checking the quality of the new application version is fully covered by the tool.
	Strongly agree	(A)
	Strongly agree	(B)
	Strongly agree	(D)
	Agree	From my point of view not the version but the tested artifact is important and only this one may be deployed. To have a comparison here would be useful. (F)
	Neutral	Lack of experience (C)
	Neutral	Does the tool check if the new version is deployed? (E)
	Neutral	The test scenarios in large projects are of course not considered - but they would have to be in Prod. deployments. Therefore possible tool but extension with testing necessary (G)
	27	I will continue to check the quality of the new application version this way.
	Strongly agree	(A)
	Strongly agree	Yeah, there's no reason not to. (D)
	Agree	(B)
	Agree	But with the artifact. (F)
	Neutral	Lack of experience (C)
	Neutral	Same here (E)
	Neutral	Depends on the project (G)
	=>	General feedback about the blue green deployment
	Very easy to use and limited to the most necessary functions. (D)	
	A blue green deployment can be easily done with the help of your tool. (E)	
	If you want to have a smooth deployment moment, you can't avoid a blue green deployment. (F)	
	The approach is good - must be extended for complex systems (G)	

Table E.6.: Results of questions regarding *EQ3* (2)

EQ	Evaluation Questionnaire Results (4) – Monitoring metrics	
EQ4	28	The metrics about the ML application are presented in a clear and understandable way.
	Strongly agree	Easy to understand (A)
	Strongly agree	There is nothing to say. (F)
	Strongly agree	All fine (G)
	Agree	Other metrics could be added too (B)
	Agree	Yes (C)
	Agree	(D)
	Agree	The metrics are very clear. (E)
	29	The metrics about the ML application are useful to better understand the ML application.
	Strongly agree	(A)
	Strongly agree	(B)
	Strongly agree	Metrics are mandatory to monitor ML applications (G)
	Agree	Yes (C)
	Agree	(D)
	Agree	Metrics are generally useful, not only for understanding (E)
	Neutral	Depends on the hypothesis under investigation. I think every project has to create its own KPIs manually. But you can start with the basic metrics. (F)
	30	I will continue to view metrics about the ML application.
	Strongly agree	Fast, simple and clear (A)
	Strongly agree	(B)
	Strongly agree	Without monitoring the metrics the quality / availability cannot be checked (C)
	Strongly agree	Certainly, I had used this function superficially before. (D)
	Strongly agree	During operation one should generally have knowledge about the metrics of applications. (E)
	Strongly agree	Necessary (G)
	Agree	These and others. (F)
	31	The metrics about the ML model are presented in a clear and understandable way.
	Strongly agree	(A)
	Strongly agree	(D)
	Strongly agree	Could be understood very well. (F)
	Agree	(B)
	Agree	Yes (C)
	Agree	as part of CloudWatch, yes (G)
	Neutral	The metrics are very clear, but not 100% understandable for someone who has no idea about ML. (E)
	32	The metrics about the ML model are useful to better understand the ML model.
	Strongly agree	(B)
	Strongly agree	(D)
	Strongly agree	Fits well here (G)
	Agree	Is easy with the existing metrics, depending on the model you need additional metrics, which you would have to add manually (A)
	Agree	Yes (C)
	Neutral	I can only partially validate with my knowledge. (E)
	Disagree	The more complicated a model is the more difficult it is to understand. Unfortunately, the metrics do not help either. Furthermore, each model is different and has different properties. (F)
	33	I will continue to view metrics about the ML model.
	Strongly agree	For a quick check and very well suited as an indicator (A)
	Strongly agree	(B)

Table E.7.: Results of questions regarding *EQ4* (1)

	Strongly agree	(D)
	Strongly agree	without metrics, no quality checks (G)
	Agree	Yes (C)
	Neutral	Possibly, yes (E)
	Neutral	Depends on the model (F)
	34	The metrics about the application infrastructure are presented in a clear and understandable way.
	Strongly agree	(A)
	Strongly agree	Simple and clearly structured, also expandable for further metrics. (D)
	Agree	(B)
	Agree	Yes (C)
	Agree	The metrics are very clear (E)
	Agree	These are sufficient for the own development. For management reporting this presentation is not sufficient. (F)
	Agree	Enough metrics to monitor this model (G)
	35	The metrics about the application infrastructure are useful to better understand the application infrastructure.
	Strongly agree	(A)
	Strongly agree	(B)
	Strongly agree	I have all information about the infrastructure at a glance. (F)
	Strongly agree	Definitely a must (G)
	Agree	(D)
	Agree	I can no longer tell which metrics are which ;) (E)
	Neutral	The metrics and the application infrastructure are only connected through scaling, only from the metrics the infrastructure cannot be identified (C)
	36	I will continue to view metrics about the application infrastructure.
	Strongly agree	(A)
	Strongly agree	(B)
	Strongly agree	Nothing to say (F)
	Strongly agree	Each of our systems has its own monitoring system for supervision (G)
	Agree	Useful for monitoring utilization / availability (C)
	Agree	(D)
	Agree	Same here (E)
	37	The automatic notification when the application infrastructure scales reaches me via the desired communication channel and is written in an understandable way.
	Strongly agree	(A)
	Strongly agree	(D)
	Strongly agree	Super Idee! Es sollte immer eine Art Benachrichtigung geben. Von selber schaut meistens keener (E)
	Strongly agree	There is nothing to say. (F)
	Agree	(B)
	Agree	Is formulated in an understandable way (C)
	Agree	Mails are ok, a dashboard application with appropriate hints would be more desirable in large systems because mails are not read :) (G)

Table E.8.: Results of questions regarding *EQ4* (2)

38	The task of automatic notification when the application infrastructure scales is fully covered.	
Strongly agree	(A)	
Strongly agree	Yes (E)	
Strongly agree	Does what it should do (F)	
Agree	Other notifications such as accuracy of trained model are important (B)	
Agree	Agree (C)	
Agree	(D)	
Neutral	Rather dashboard application with corresponding hints (G)	
39	I will continue to rely on notifications when the application infrastructure scales.	
Strongly agree	(A)	
Strongly agree	(B)	
Strongly agree	There should always be notifications! (E)	
Agree	Is very useful. (D)	
Agree	Yes but other notifications (G)	
Neutral	Possibly notifications or even just monitoring via dashboards or similar (C)	
Neutral	The notification is interesting for the company. (F)	
=>	Please provide general feedback about viewing the metrics.	
Very helpful and efficient! (D		
As I mentioned above, this is a few too many questions about metrics. These should either be specified more precisely, or summarized (E)		
The metrics for the infrastructure can be used very well as a template. For models, the metrics have to be adjusted by hand. (F)		
Metrics are an extremely important topic and must be considered (G)		

Table E.9.: Results of questions regarding *EQ4* (3)

8. Bibliography

[1] Radon, https://pypi.org/project/radon/

[2] Arnold, M., Boston, J., Desmond, M., Duesterwald, E., Elder, B., Murthi, A., Navratil, J., Reimer, D.: Towards Automating the AI Operations Lifecycle (mar 2020)

[3] Atwal, H.: Practical DataOps. Apress (2020)

[4] AWS: Amazon EC2, https://aws.amazon.com/ec2/

[5] AWS: Amazon ECS - Run containerized applications in production, https://aws.amazon.com/ecs/

[6] AWS: Amazon Virtual Private Cloud (VPC), https://aws.amazon.com/vpc/

[7] AWS: CloudFormation, https://aws.amazon.com/cloudformation/

[8] AWS: Elastic Container Registry (ECR), https://aws.amazon.com/ecr/

[9] AWS: VPC with public and private subnets (NAT) - Amazon Virtual Private Cloud, https://docs.aws.amazon.com/vpc/latest/userguide/VPC{_}Scenario2.html

[10] AWS: What Is an Application Load Balancer? - Elastic Load Balancing, https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

[11] AWS: Übersicht über Amazon Web Services. Tech. rep. (2018)

[12] AWS: Amazon EKS - Managed Kubernetes Service (2019), https://aws.amazon.com/eks/

[13] AWS: Elastic Load Balancing - Amazon Web Services (2019), https://aws.amazon.com/elasticloadbalancing/

[14] Bamboo Continuous Integration and Deployment Build Server: https://www.atlassian.com/software/bamboo

[15] Baylor, D., Breck, E., Cheng, H.T., Fiedel, N., Foo, C.Y., Haque, Z., Haykal, S., Ispir, M., Jain, V., Koc, L., Koo, C.Y., Lew, L., Mewald, C.: TFX: A TensorFlow-based production-scale machine learning platform. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. vol. Part F1296, pp. 1387–1395. Association for Computing Machinery, New York, USA (aug 2017)

[16] Bosch, J., Crnkovic, I., Olsson, H.H.: Engineering AI Systems: A Research Agenda (jan 2020)

[17] Breck, E., Cai, S., Nielsen, E., Salib, M., Sculley, D.: The ML test score: A rubric for ML production readiness and technical debt reduction. In: Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017. vol. 2018-Janua, pp. 1123–1132 (2017)

[18] CloudFoundry: https://www.cloudfoundry.org/

[19] CloudWatch - Application and Infrastructure Monitoring: https://aws.amazon.com/cloudwatch/?nc1=h{_}ls

[20] David, W., Brandt, P.: Rules of Machine Learning: — ML Universal Guides — Google Developers, https://developers.google.com/machine-learning/guides/rules-of-ml

[21] Davis, F.D.: A technology acceptance model for empirically testing new end-user information systems : theory and results. Tech. rep. (1985)

[22] Doebel, I., Leis, M., Vogelsang, M., Neustroeov, D., Petzka, H., Rueping, S., Voss, A., Wegele, M., Welz, J.: Maschinelles Lernen - Kompetenzen, Anwendungen und Forschungsbedarf (2018)

[23] Doebel, I., Leis, M., Vogelsang, M., Neustroeov, D., Petzka, H., Rueping, S., Voss, A., Wegele, M., Welz, J.: Maschinelles Lernen. Eine Analyse zu Kompetenzen, Forschung und Anwendung (2018)

[24] Ernst, D., Becker, A., Tai, S.: Rapid Canary Assessment Through Proxying and Two-Stage Load Balancing. In: Proceedings - 2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019. pp. 116–122. Institute of Electrical and Electronics Engineers Inc. (may 2019)

[25] GitHub - concourse/concourse: Concourse is a container-based continuous thing-doer written in Go and Elm.: https://github.com/concourse/concourse

[26] GitHub - Netflix/metaflow: Build and manage real-life data science projects with ease.: https://github.com/Netflix/metaflow

[27] Gravity Overview - Gravitational Gravity: https://gravitational.com/gravity/docs/

[28] HasiCorp: TerraForm, https://www.terraform.io/

[29] Jalali, S., Wohlin, C.: Systematic literature studies: Database searches vs. backward snowballing. In: International Symposium on Empirical Software Engineering and Measurement. pp. 29–38. ACM Press, New York, USA (2012)

[30] Jeff, N.: Docker in Action. Manning Publications, 2 edn. (2019)

[31] Johnston, J.: Docker in the Trenches: Successful Production Deployment. Bleeding Edge Press, 1 (early release) edn. (2015)

[32] Kibana: Visualisieren, Analysieren und Erkunden von Daten — Elastic: https://www.elastic.co/de/kibana

[33] Kleebaum, A., Johanssen, J.O., Paech, B., Bern, B.: Continuous Management of Requirement Decisions Using the ConDec Tools - heiDOK (2020)

[34] Lexico: Key Performance Indicator, https://www.lexico.com/definition/key{_}performance{_}indicator

[35] Logstash: Collect, Parse, Transform Logs — Elastic: https://www.elastic.co/logstash

[36] Lwakatare, L.E., Raj, A., Bosch, J., Olsson, H.H., Crnkovic, I.: A taxonomy of software engineering challenges for machine learning systems: An empirical investigation. In: Lecture Notes in Business Information Processing. vol. 355, pp. 227–243. Springer Verlag (2019)

[37] Mell, P.M., Grance, T.: The NIST definition of cloud computing. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD (2011), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[38] MHP: Artificial Intelligence — MHP – A Porsche Company, https://www.mhp.com/de/services/focus-topics/artificial-intelligence{#}!artificial-intelligence-driving-revolution-

[39] Mitchell, T.: Machine Learning (1997)

[40] Muthusamy, V., Slominski, A.: Towards enterprise-ready AI deployments Minimizing the risk of consuming AI models in business applications. Tech. rep. (2019)

[41] Pethuru, R.: Learning Docker. Packt Publishing (2015)

[42] Rountree, D., Castrillo, I.: Introduction to the Cloud. In: The Basics of Cloud Computing, pp. 1–17. Elsevier (jan 2014)

[43] Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2), 131–164 (apr 2009)

[44] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (2020)

[45] Sato, D., Wider, A., Windheuser, C.: Continuous Delivery for Machine Learning (September), 1–34 (2019), https://martinfowler.com/articles/cd4ml.html

[46] Splunk Platform — Splunk: https://www.splunk.com/en{_}us/platform.html

[47] Studer, S., Bui, T.B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., Mueller, K.R.: Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology (mar 2020)

[48] What is Elasticsearch? — Elastic: https://www.elastic.co/de/what-is/elasticsearch

[49] Wittig, A.: Amazon web services in action (2015)

[50] Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: ACM International Conference Proceeding Series. pp. 1–10. Association for Computing Machinery, New York, New York, USA (2014)

[51] Xu, Y., Chen, N., Fernandez, A., Sinno, O., Bhasin, A.: From infrastructure to culture: A/B testing challenges in large scale social networks. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. vol. 2015-August, pp. 2227–2236. Association for Computing Machinery, New York, USA (aug 2015)

[52] Young, S.W.H.: Improving Library User Experience with A/B Testing: Principles and Process. Weave: Journal of Library User Experience 1(1) (aug 2014)

[53] Zhang, J.M., Harman, M., Ma, L., Liu, Y.: Machine Learning Testing: Survey, Landscapes and Horizons (jun 2019)