RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

MASTER THESIS

Platform to Assist Medical Experts in Training, Application, and Control of Machine Learning Models Using Patient Data from a Clinical Information System

Author:
Matthias GREINER

Supervisors:
Prof. Barbara PAECH
Prof. Artur ANDRZEJAK
Anja KLEEBAUM
PD Dr. Klaus MAIER-HEIN
Dr. Marco NOLDEN
Jasmin METZGER

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Software Engineering Group
Applied Computer Science

2019-08-20

Declaration of Authorship

I, Matthias GREINER, declare that this thesis titled, “Platform to Assist Medical Experts in Training, Application, and Control of Machine Learning Models Using Patient Data from a Clinical Information System” 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:

Abstract

Platform to Assist Medical Experts in Training, Application, and Control of Machine Learning Models Using Patient Data from a Clinical Information System

In recent years, clinical data scientists achieved major breakthroughs advancing machine learning models for the medical domain, which have great potential assisting medical experts. Machine learning models can be leveraged to assist medical experts in tasks such as analyzing and diagnosing patient data, for example, from computed tomography scans. However, it is a challenge to translate the latest advancements in academic research fields such as computer sciences and physics into clinical practice. For this purpose, clinical data scientists and medical experts need to closely collaborate. This thesis tackles challenges of accessibility and usability of state-of-the-art machine learning models as well as designing a scalable computing architecture. Hence, conceptual ideas of possible strategies, as well as a prototype of such a machine learning platform, are presented.

A systematic literature review was conducted on the current approaches to create medical machine learning platforms, the management of machine learning models, and the version management of large data sets. Afterward, the functional and non-functional requirements of the new machine learning platform were elicited as part of the requirements analysis. Two streamlined workflows for clinical data scientists and medical experts were derived from the requirement analysis. The workflow for the clinical data scientists includes steps to define, train, and share machine learning methods, including pre- and postprocessing modules, and management of data sets. Medical experts are able to analyze patient data using pre-defined machine learning methods. Building on the result of these analyses, the architecture of the platform was derived. The architecture consists of a scalable infrastructure stack, a lightweight and easy-to-use web interface, as well as a backend component to provide the required functionalities. The final design decisions solve the issue of efficiently standardizing, parallelizing, and applying machine learning workflows within a scalable computing infrastructure.

The proposed platform was evaluated with 22 participants, consisting of clinical data scientists (N=12) and medical experts (N=10). Both groups were asked to rate specific workflows of the platform, as well as the platform as a whole, and to provide additional ideas and feedback. 92% of the medical experts and 90% of the clinical data scientists rated their overall impression of the platform as *very good*. Furthermore, medical experts and clinical data scientists *strongly agreed* that the platform facilitates method development and collaborations with 92% and 90%, respectively.

The conducted expert survey suggests that the here proposed platform could be used to develop, optimize, and apply machine learning methods in the medical domain and beyond, thereby easing the collaboration between medical experts and clinical data scientists.
The prototypical implementation is published on GitHub, accessible using the following link: https://github.com/magreiner/MMLP.

Zusammenfassung:

Klinische Datenwissenschaftler haben in den letzten Jahren große Durchbrüche im Bereich des medizinischen maschinellen Lernens erzielen können. Diese Durchbrüche haben großes Potential, Mediziner in ihrer Arbeit zu unterstützen. Beispielsweise können Algorithmen des maschinellen Lernens genutzt werden, um Mediziner bei der Analyse von Patientendaten zu unterstützen. Eine Beispielanwendung ist die Analyse von computertomographischen Bilddaten. Jedoch ist es eine Herausforderung, den neusten Stand der akademischen Forschung in Natur- und Computerwissenschaften in die klinische Praxis zu überführen; eine enge Zusammenarbeit zwischen Datenwissenschaftlern und Medizinern ist daher essentiell. Das Ziel dieser Arbeit ist es, die Entwicklung, Zugänglichkeit und Anwendbarkeit modernster Algorithmen des maschinellen Lernens zu verbessern. Dafür wird in dieser Arbeit eine Plattform entworfen, die Mediziner und Datenwissenschaftler im Bereich des maschinellen Lernens im klinischen Kontext unterstützt. In der Arbeit werden Anforderungen, mögliche Lösungsansätze, sowie ein Prototyp einer solchen Plattform vorgestellt.

Als Grundlage wurde eine systematische Literaturrecherche zu aktuellen Lösungsansätzen durchgeführt. Dabei werden Strategien zur Erstellung einer solchen Plattform, zur Verwaltung von Algorithmen des maschinellen Lernens, sowie für die Versionierung und Datenverwaltung von sehr großen Datensätzen untersucht. Anschließend werden im Rahmen der Anforderungsanalyse funktionale und nichtfunktionale Anforderungen an die Plattform ermittelt. Der Arbeitsablauf eines klinischen Datenwissenschaftlers umfasst das Definieren, Trainieren, und das Teilen der Algorithmen des Maschinellen Lernens. Wohingegen der Fokus des Arbeitsablaufs für Mediziner darin liegt, neue Patientendaten mithilfe der zuvor erstellen Algorithmen in der Plattform zu analysieren. Die aus den Anforderungen abgeleitete Architektur der Plattform beinhaltet eine skalierbare Infrastruktur, eine leichtgewichtige und benutzerfreundliche Web-Schnittstelle, sowie ein Backend, welches die Funktionen der Plattform umsetzt.

Die vorgeschlagene Plattform wurde mit 22 Teilnehmern evaluiert, die sich aus klinischen Datenwissenschaftlern (N=12) und medizinischen Experten (N=10) zusammensetzten. Die Teilnehmer wurden gebeten, sowohl bestimmte Prozesse innerhalb der Plattform, als auch die Plattform als Ganzes zu bewerten. Des Weiteren wurden die Teilnehmer gebeten weitere Verbesserungsvorschläge, Rückmeldungen und Anmerkungen zu geben. 92% der Mediziner und 90% der Datenwissenschaftler bewerteten ihren Gesamteindruck der Plattform als *sehr gut*. Darüber hinaus äußerten 92% der Mediziner und 90% der Datenwissenschaftler eine *starke Zustimmung* zu der Aussage, dass die Plattform die Methodenentwicklung und Zusammenarbeit erleichtert.

Die durchgeführte Evaluierung legt nahe, dass die in dieser Arbeit erstellte Plattform zur Entwicklung, Optimierung und Anwendung von Algorithmen des maschinellen Lernens im medizinischen Kontext verwendet werden und somit insbesondere die Zusammenarbeit zwischen Datenwissenschaftlern und Medizinern erleichtern könnte.
Die prototypische Implementierung wurde auf GitHub, unter dem Link: https://github.com/magreiner/MMLP, veröffentlicht.

Acknowledgements

Throughout the writing of my master thesis, I have received a great deal of support and assistance.

Foremost I would like to thank my supervisors Prof. B. Paech, Prof. A. Andrzejak, Dr. K. Maier-Hein, and Dr. M. Nolden, who spend much effort to make it possible to write a thesis in the medical context of the German Cancer Research Center Heidelberg (DKFZ).

I would especially like to acknowledge my direct contacts Anja Kleebaum, Jasmin Metzger, and Dr. Marco Nolden, for all the hours they invested in me and my thesis, always trying to answer my questions quickly, and all the interesting meetings we had. Thank you for listening to my ideas, and all the advice and support you gave me to conduct my research, I was able to learn a lot about academic and scientific work.

In addition, I would like to give a special thank you to my friends Venelin Petkov, Bünyamin Pekdemir, and Dr. Mohammadreza Ghanavati, for providing me with exceptional technical and emotional support. Your support enabled me to achieve more powerful and sophisticated results.

Further, I want to thank my colleagues from the DKFZ, especially Gregor Ramien, Sebastian Gruber, Farhan Rasheed, Hafiz Fahad, Tobias Stein, and Silvia Seidlitz. Thank you for all the support, especially during more difficult times, and the great time we had together.

I am also grateful to each of the various testers of my platform and the 22 participants of my evaluation. I really appreciate that you spend so much time to support my thesis, even when your schedule was already very tight. Thank you especially for all the ideas and feedback you provided, it helped me a lot to improve my results.

Besides I want to thank the people working in the background, such as the administrative and technical staff from the University Heidelberg, especially Frau Anke Sopka, and from the DKFZ, especially Michaela Gelz, Janina Dunning, and Stefanie Strzysch.

Finally, I would like to express my very profound gratitude to my family for all their unfailing support, wise counsel, and sympathetic ear throughout my studies and my life in general. Especially while writing the thesis it helped me a lot to always have someone to talk to and find solutions I did not see immediately.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Domain-Specific Challenges

Machine learning models can be leveraged to assist medical experts in various tasks such as analyzing and diagnosing patient-related data such as images, video, audio, and text. These models are used to teach a computer to iteratively improve on a specific set of tasks, such as detecting areas of interest in a large number of patient images. The results are compared with the real area of interest, the *ground truth* annotation created by a medical professional. The differences between the output of the models and the ground truth annotation are used to update the models, and the entire process is repeated until the prediction accuracy is sufficient. In the medical field, machine learning models are, for example, used for segmentation of patient images. Segmentation approaches are able to predict if a tumor presence and location in a patient image.

Working with patient data requires a large amount of storage and computation power. Therefore, distributed cloud-native computing environments such as Kubernetes [18] and Apache Mesos [42] are frequently leveraged. Furthermore, they provide medical experts with the ability to scale compute resources as needed. These environments can manage application containers such as Linux Containers [99] and Docker [16] across multiple computing machines. Application containers enable the use of various tools such as machine learning and services required to operate the machine learning platform. Such service containers are utilized to add new components to the platform. Since multiple clinical data scientists may collaborate in developing state-of-the-art custom machine learning solutions, the concept should allow parallel work, training, and inference of machine learning models.

The application of state-of-the-art machine learning models in the medical field goes along with significant challenges. Sometimes it can take a long time before machine learning models are applied in the clinical research environment. One explanation for this could be that there are no simple ways to share these models. Further, the medical expert may feel overwhelmed by technical details while trying to apply state-of-the-art machine learning models on their patient data. The clinical data scientist, who is not a domain expert, may work in the meantime on other projects. Therefore, the developed machine learning model is not maintained or finalized, as it is not applied in medical practice. On the other side, the medical expert may not know that new models exist that could help with her work. If later they want to use the model, the developer might not be available anymore. Much knowledge could be lost, which makes it much more difficult to reactivate the model, especially if the hardware and software versions changed and the source code needs to be rewritten or the model needs to be re-trained.

In summary, extensive knowledge in the machine learning domain and the medical area is required to achieve the meaningful application of models, making collaboration between experts in different fields essential. Medical experts have little knowledge of how to utilize the models created by the clinical data scientist. Since data scientists are focussing on solving crucial problems, they greatly benefit from the input of the first-hand experience from the medical domain, thus nurturing tailored solutions. Therefore, both parties would greatly benefit from a standardized platform to support developing and applying models, enabling the broad application.

1.2 Research Methodology

In this thesis, we aim to follow the approach described in the book Design Science Methodology for Information Systems and Software Engineering [93]. To solve the problem at hand, we divide the tasks into the following four parts:

1.2.1 Part 1: Research Problem

In the first part, the research problem is analyzed. The research problem of this thesis is the design of a platform to assist medical experts in training, application, and control of machine learning models using patient data from a clinical information system.

1.2.2 Part 2: Problem Investigation

In the second part, we aim to learn more about the state-of-the-art and state-of-the-practice of machine learning in the medical context. The resulting research questions from the first part are investigated in two steps. In the first step, we conducted an extensive literature review, in which we searched multiple scientific databases to find matching publications. On these publications, we applied snowballing techniques to additionally find and analyze related research literature in this field. To further gain information about the medical context, experts in the field and potential users of the potential platform were interviewed, using contextual and one-on-one interviews.

1.2.3 Part 3: Treatment Design

Based on the knowledge collected in the problem investigation stage, we finalized the requirements analysis. We then propose a design and a prototype of a platform to tackle these requirements.

1.2.4 Part 4: Treatment Validation

In the last part, we ask potential users to evaluate the proposed platform. This completes the design cycle, and we discuss the results of the evaluation, with the focus on how good our proposed platform, i.e. the treatment design, solved the researched problem solved the beforehand introduced researched problem. This could be the starting point of a new design cycle to improve the platform further.

1.3 Goals and Contributions

The main goals of the thesis are summarized in Table 1.1. Goal *GO-2* is more precisely specified by defining the required design goals shown in Table 1.2 to achieve it. The

TABLE 1.1: Goals of this thesis

Goal	Description
GO-1	Overview of state-of-the-art by conducting a systematic literature review and overview of state-of-the-practice by performing interviews and field studies
GO-2	Requirements analysis, design and implementation of a prototypical medical machine learning platform
GO-3	Evaluation of the platform

TABLE 1.2: Design goals required to achieve goal GO-2

Design Goals	Description
DG-1	Support for viewing, storing, sharing, importing, and training of machine learning models
DG-2	Support for analyzing patient data using machine learning methods
DG-3	Storage service for various types of custom data sets
DG-4	Export of newly generated predictions and training results
DG-5	Integrability into a medical distributed computing environment

overall goal is to simplify and therefore, further integrate the application of models into the daily work of medical experts and clinical data scientists. The first goal *GO-1* is to perform a systematic literature research to examine current state-of-the-art and state-of-the-practice in the context of a medical machine leaning platform. The second goal *GO-2*, yields a requirements analysis, a design, and a prototypical platform as the treatment described in Section 1.2. In addition, the approach should be designed in a way that automates many of the necessary steps of machine learning and works in an iterative fashion, where models are trained on annotated medical data and then inferred on non-annotated medical data. Finally, the third goal *GO-3* is to evaluate and discuss the envisioned platform.

The major contributions of this thesis are the thorough accomplishment of beforehand mentioned goals. Additionally, we plan to *open-source the thesis and additional resources* to allow the community to use and further develop the proposed platform.

The novel contribution of the platform is the combined ability to provide a comprehensive feature set with almost no restrictions, in, e.g., machine learning frameworks or data types, while offering a standardized way to share models as a packaged, out-of-the-box functional method within an easy-to-use webinterface. Notably, the frontend, backend, computing environment, and infrastructure stack are designed with scalability in mind, resulting in a future-proof architecture of core components of the platform.

Therefore, the proposed platform can provide the foundation for a productive collaboration hub for data scientists and medical experts.

1.4 Outline

This thesis is structured in the following way: The **Introduction** chapter describes the motivation of this work, the goals, the research methodology, and the outline of this thesis. In the **Background** chapter, the context of this work, i.e., machine learning techniques, as well as infrastructure and cloud-native development approaches, are explained. It is followed by the **Literature Research** chapter, which reviews current approaches to apply machine learning techniques in the medical field and seeking answers for multiple research questions. Next the **Requirements Analysis** chapter defines which requirements are needed for the envisioned platform. After defining the requirements, the **Design and Implementation** chapter presents the envisioned platform, its components, and interfaces. Within the **Quality Assurance** chapter, a plan on how to ensure the quality of the platform is focussed. The **Evaluation** chapter explicate how this work is evaluated and discusses the results. In the last chapter, **Conclusion and Future Directions**, the content of this thesis and how this work is evaluated and discusses the results is summarized.

Chapter 2

Background

The background chapter describes the theoretical concepts on which this thesis builds upon. Section 2.1 deals with differences between strategies to isolate applications running on a single machine. Three different isolation strategies are discussed to support the design of a larger platform. The larger platform builds on a (cloud) compute cluster, which is explained in detail in Section 2.2. The way of designing cloud-native applications and services to work efficiently on larger platforms is explained in Section 2.3. This thesis aims to virtualize machine learning approaches. Therefore, machine learning is introduced in Section 2.4.

2.1 Program Isolation Strategies

Program isolation is necessary to ensure that multiple programs are able to run on a machine simultaneously, without interfering with each other. There are various methods for isolating programs on a machine [82]. Non-virtualized environments provide direct access to the underlying hardware, which is often used in traditional setups, as shown in Figure 2.1(a). This may be a significant advantage for applications requiring direct hardware access but poses lots of challenges such as scalability, handling of resource heterogeneity, incompatible dependencies, efficiency, and cost for diverse applications.

2.1.1 Hardware-Level Virtualization (Hypervisor)

Another approach is to use hardware-level virtualization to encapsulate applications into separate virtual machines. Hardware-Level virtualization adds an additional hardware abstraction layer, which allows running multiple virtual operating systems on a single physical machine. Each virtual machine has its own virtual hardware resources such as CPU, memory, storage, as well as networking and is completely isolated from other virtual machines. Further advantages are better scalability and higher efficiency, which leads to lower cost of the system.

Virtualization was introduced in the early ’70s to improve sharing and utilization of expensive resources. The core of this technique is an abstraction layer, the hypervisor, which is added on top of the bare metal hardware (or alternatively on top of the host operating system) and handles the connection between the hardware and the virtual machines, also known as full hardware virtualization. With the hypervisor, it is possible to run multiple independent operating systems (*guests*) on the same

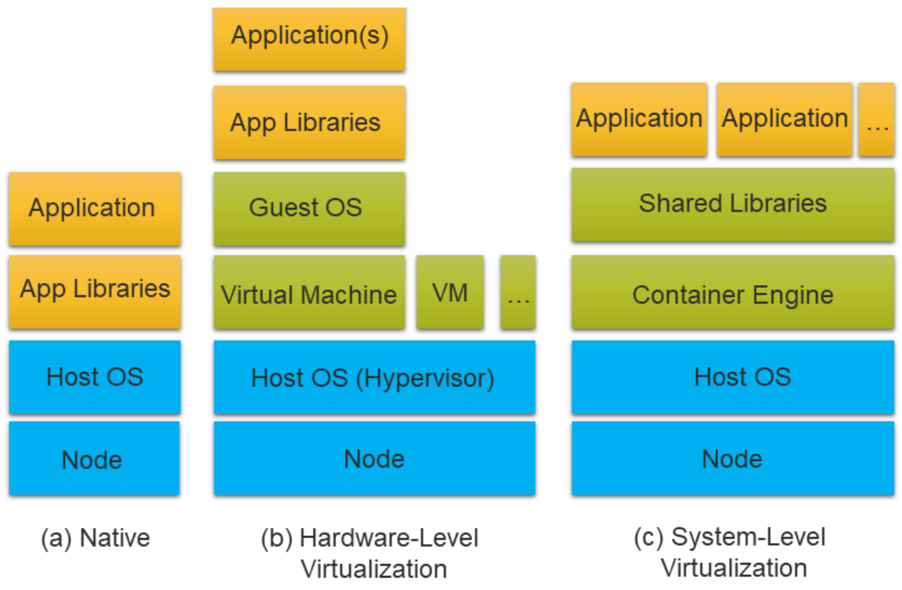


FIGURE 2.1: Overview over different virtualization concepts

physical machine (*host*) without any interference between them [61]. The architecture of full hardware virtualization can be seen in Figure 2.1(b).

When using hardware virtualization, the guest operating system is encapsulated in a *virtual machine* (VM) on the host machine. It defines the virtual hardware the guest system can use such as CPU cores, amount of memory, and network interfaces. This may increase efficiency since virtual machines can share the resources from a host. Virtual machines support a lot of additional features compared to physical machines [61], which are described in the following paragraphs.

Consolidation: Various services running on different operating systems can each be installed inside a virtual machine and run on the same physical machine. With multiple services running on the same machine, resources such as CPU cores, memory, and network can be multiplexed and shared between them. This is a significant advantage assuming that different services on a host will not show peak usage of the same resource at the same time.

Isolation: Each virtual machine is independent of other virtual machines, even from those on the same host. It allows increased reliability and availability for multiple services, since a problem in one virtual machine may not affect other virtual machines.

Virtual Environment: The state of the virtual machine can be exported to a file (called snapshot), even during runtime. It can be used to, for example, debug a service, without having to use a separate development machine, since the virtual environment can be replicated and used multiple times This feature is especially useful to archive the changes made during runtime, for example, archiving the results of a machine learning model training pipeline.

Scalability: Changing of service demand often requires to adjust the allocated resources to ensure an excellent service experience and effective resource utilization.

Independence: Since each virtual machine is defined via software, it can run on any physical machine supporting virtualization.

Unfortunately, there are still many drawbacks in terms of application performance, latency, portability, and life-cycle management [32]. These drawbacks are especially severe for compute-intensive tasks such as training of machine learning models.

2.1.2 System-Level Virtualization (Application Container)

Lightweight process encapsulation (containerization) tackles the issues mentioned above. Instead of running the application inside its own virtual operating system, only the application itself is deployed, inside its own virtual environment, sharing the host operating system kernel. Similar ideas were already implemented in Solaris Zones, BSD jails, AIX Workload Partitions, and Linux-based containers projects [77]. The support for namespaces (process isolation) and cgroups (resource management) were added to the Linux Kernel in version 3.8 (February 2013), which made it possible to encapsulate processes on a system-level instead of using fully virtualized operating systems.

This approach was developed further to application containers, which allows encapsulating an entire application, including its dependencies [85]. Instead of running a full operating system inside each virtual machine, which decreases performance and increases resource usage, containers can access the host's kernel directly. Additionally, libraries which are needed by multiple applications can be shared globally, instead of replicating them into each virtual environment.

These changes result in much faster start-up times of applications, add nearly no overhead on processing and storage usage and are more portable, compared to virtual machines. The architecture of software-level virtualization using application containers can be seen in Figure 2.1(c).

The most common solutions are Linux Containers [77] (LXC, set of userspace tools and utilities – used by Docker and LXD [82]), OpenVZ [99, 52] (used for providing host and cloud services), and Linux-VServer (soft partitioning concept based on Security Contexts) [85].

2.2 Cloud Compute Cluster Fundamentals

Machine learning-related tasks may require substantial computing resources, exceeding the processing power of a single workstation and GPU. Therefore, it may be advantageous to use a scalable compute cluster as a foundation for the environment platform. A compute cluster infrastructure consists of multiple workstations, connected to accumulate the available resources for larger tasks, as shown in Figure 2.2. The infrastructure can be installed locally, dynamically rented from a cloud provider, or partly installed locally combined with resources from a cloud provider. Fully running the computing environment in the cloud is the trending solution since it offers much more possibilities, higher efficiency, and lower cost, but also faces significant challenges in terms of data security and privacy, especially in the medical domain.

Infrastructure as a Service (IaaS) is the most basic service model in the cloud. It does offer a virtual place, where the tenant can freely select and configure IT resources of nearly any kind such as computing nodes, networking parts, security policies, other services such as databases and elastic storage. The provider will only provide the services / hardware selected without managing or regulating the usage.

***Compute Cluster* as a Service (CaaS)** is a service managed by the cloud provider which offers a fully operational compute cluster. The tenant has access to the managing console and tailors the cluster configuration to his needs. Since the cloud provider manages the cluster, the user has much fewer customization options but does not have to spend time on updating and maintaining the cluster itself.

The most popular cloud providers to offer IaaS and CaaS are Amazon Web Services [96], Google Cloud Platform [54] and Microsoft Azure [94].

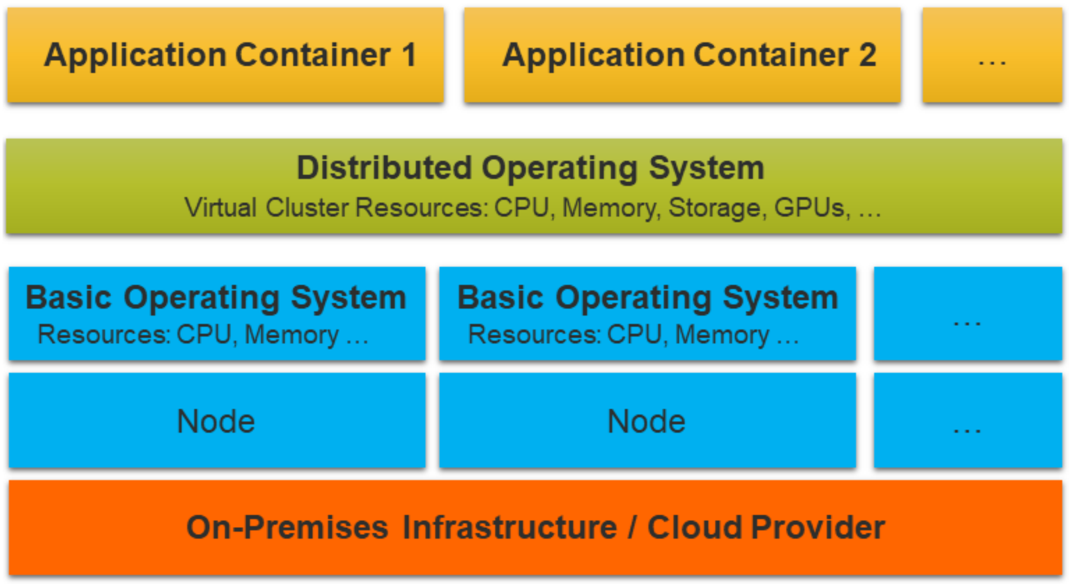

Application Container 1
Application Container 2
...
Distributed Operating System
Virtual Cluster Resources: CPU, Memory, Storage, GPUs, ...
Basic Operating System
Resources: CPU, Memory ...
Basic Operating System
Resources: CPU, Memory ...
...
Node
Node
...
On-Premises Infrastructure / Cloud Provider

FIGURE 2.2: Generic cloud computing cluster architecture

2.2.1 Provisioning the Infrastructure

The open-source cloud operating system OpenStack [79] is frequently used for on-premises cloud offerings. OpenStack provides cloud computing functionalities on top of large pools of commodity hardware such as compute resources, GPUs, storage, and networking. It was founded in July 2010 by Rackspace Hosting and NASA and can now be seen as the de-facto open standard cloud computing platform.

2.2.2 Base Operating System

The first actual steps to create a cluster is to set up the single nodes and prepare them to be integrated into a larger system. Within the single node, a specialized operating system is used, which should be as minimal and resource-friendly as possible. This is essential because the system is always running and lays the groundwork for all functionality of the cluster. The base operating system should be battle-proven (really stable), provide a safe way to apply updates fast, include a fail-over strategy, consumes a minimal amount of resources and increased security for task isolation and protect against all kinds of attacks.

Therefore a modern, lightweight operating system is beneficial, which is designed primarily for running containerized applications as a base of a large cluster. CoreOS [66] is such a base operating system. CoreOS is a small Linux system that only includes a minimal set of tools and the essential tools to run Docker [16] containers. There is no package manager since the applications can be installed by downloading and running a container. The system size and resource consumption profits from the minimal system, so that the running applications show increased performance.

2.2.3 Distributed Operating System

The basic operating system manages one single node and its resources. To connect the nodes and aggregate over their resources, a distributed operating system is needed. Two of the most popular ones are Kubernetes [13] and Apache Mesos [42, 46], which are currently used by a lot of big companies such as Google, Twitter, and Facebook.

Kubernetes runs on top of the base operating system of each node and integrates many features into a large cluster managing platform. It adds a layer on top of the nodes, which combines their resources to a single resource pool, and allows containers to allocate and use the resources of the desired node. The most important features are a distributed operating system, resource manager, and a container orchestration platform. Application container can use features such as high availability deployment, service discovery within the namespace, and advanced service health monitoring and control.

2.2.4 Containerized Applications

There are multiple ways to isolate applications on an operating-system-level. One approach is to use container environments such as Docker. Docker [16] is a popular containerizer, which is supported by most cloud providers. It is based on Linux Containers (LXC), open-source and provides a rich tool-set to work with containers. LXC is a set of user-space and utilities to encapsulate applications, including their dependencies. Most Linux based systems support LXC, which makes it possible to run the same application container on multiple systems. Additionally, Docker provides tools for a fast build process and to slimline application definitions to be able to distribute them in a fast and consistent way.

The main advantage is the newly introduced *Dockerfile*, which is the core of a Docker container. Instead of saving the whole virtual environment, Docker separates the actual application code from the actual runtime-data. Usually, the application code is much smaller than the full environment, which is a significant advantage for saving, sharing, and debugging containers. The application code defines what will happen after the Docker container is started, containing all the installation and configuration procedures. Docker provides three tools to run, build, and control containers.

Docker Daemon provides the docker container functionality. It is running in the background and manages the running containers. The Docker client can be used to control its behavior and monitor the processes.

Docker Client provides the command-line tool to access, build, and manage Docker container. It communicates with the Docker daemon to be able to fulfill the requests.

Docker Registry stores and manages Docker container images. Each Docker container uses a basic system image, on top of which the commands from the Dockerfile are executed. Every command from the Dockerfile creates a new layer on the image, which can be stored in a new image and used as a base image in another container. This architecture behaves similar to an onion, having the base image in the core and adding the command-layers on top of it.

2.3 Cloud-Native Services Approach

Cloud-native services are used to perform specialized tasks, such as providing the background functionality required for larger websites. In general, services are running in the background and await requests from users or other services. In this work, services should provide cloud-native functionality. According to [95], the ten critical factors to make a service cloud-native are the following:

1. Encapsulated into a lightweight virtual environment
   All components and requirements for the service are well defined and running in a lightweight virtual environment. The environment is usually based on a system-level application container. Multiple containerizers are available. The most common containerizer is Docker [16]. These containers can scale rapidly, based on the current demand of the service and also increase infrastructure utilization.

2. Optimized programming language and framework for each service
   Large monolithic are written in a single programming language or framework. Due to the tightly coupled structure of these services, it is not possible to use the best-suited language or framework for each task. Containers are very flexible and capable of running multiple services, using different programming languages easily. Therefore it is possible to optimize the tasks by deploying them into their own container and using the optimal environment to fulfill the desired job.

3. Loose coupling between services
   A larger service often requires multiple smaller functionalities. By combining

small functions, a more sophisticated service can handle more complex tasks. Decoupling of the smaller functions into separate containers help to optimize each function, maintain them independently, and share them with other services which use them. Scaling is possible on a very fine-graded level, further reducing bottlenecks of broader services.

4. APIs for interaction with the services
   The communication between the containers are based on protocols such as the representational state transfer (REST) [9] and remote procedure calls (RPC) [81]. This simplifies and standardizes how the components communicate with each other and allow single components to be easily replaced by others.

5. Separation of stateless and stateful services
   The stateful service stores information of each request and customizes the functionality accordingly. The ideal service, on the other hand, behaves stateless, thus does not retain session information or status of a request. Stateless services are much easier to maintain and scale since the stored information do not need to be considered.

6. Isolated from the operating system and hardware
   Hardware components and operating systems can vary depending on various factors. Cloud-native services run in virtualized environments and do not rely on specific hardware or software components. However, it is possible to reserve specific resources, e.g., storage or the number of CPU cores. Additionally, specific components may be required, such as solid-state drives and graphics processing units (GPUs).

7. Deployed in a cloud environment
   The infrastructure for cloud-native applications is virtualized, shared between multiple users.

8. Leveraging common DevOps tools for managing the services
   Continues integration and delivery pipelines are usually managing services, reducing the workload of the programmer.

9. Advanced automation
   Automation plays an essential role for cloud-native application. They can scale based on the current demand, are re-deployed after code changes. Multiple environments are used to test changes and run services for production.

10. On-demand resource allocation and scaling
    Due to the usage of virtual environments, it is possible to monitor and scale small functions. By defining rules on the availability and the responsibility, the services can scale themselves based on their current needs.

2.4 Machine Learning

In traditional computing, a specific problem is solved by defining exactly what the computer should do. However, machine learning uses another approach. Instead of defining exactly what the computer should do, the developer aims to create a generic solution, and iteratively tries to improve the outcome of the generic approach until the problem is solved sufficiently.

This generic solution is called a machine learning model. The model is represented as an algorithm with a high number of of variables. Depending on the values of the variables (called parameters), the outcome changes. Therefore the developer aims to (1) find a good architecture to solve this problem and (2) to find a set of parameters which lead to a good outcome. The process of creating a model consists of obtaining a data set, choosing a network architecture tailored to the problem and training this architecture on the selected data set. Choosing a specific network architecture matching the task and the data set is crucial. Depending on the task at hand, many different architecture types are available. In the case of computer vision, for example, often convolutional neural networks are chosen, which have been very successful [55].

Training is commonly done either supervised or unsupervised. Supervised refers to the process of showing the network an example, along with a label corresponding to the desired network prediction. Iteratively, as more examples are shown the parameters converge towards values that are able to describe the presented examples, but in theory, are also able to make reasonable predictions on unseen data. Unsupervised training does the same, but will not show the correct label alongside the training examples, but another optimization goal is defined, that tunes the network to the desired outcome. This can refer to various goals, such as merely finding clusters within the data. The process of using the network to make predictions on unseen data is called inference or application. For this process, the network parameters are frozen, and the input is fed into the model. The network then makes a prediction according to its prior training.

To simplify these tasks of finding the matching network architecture and optimizing the parameters for specific tasks, various frameworks offer abstractions and specialized functionalities. Popular machine learning frameworks are Tensorflow [5] and Pytorch [75]. Each framework has its own requirements and specifics, making it very difficult to generalize the environment to support multiple frameworks.

Leveraging machine learning for computer-aided analysis of medical data dates back more than 20 years [58]. Nowadays, it is used in the clinical workflow at various stages from population screening and diagnosis to treatment delivery and monitoring [34].

Machine learning models are, for example, used for semantic segmentation, instance detection, and instance segmentation. Semantic segmentation predicts each pixel in a patient image, such as tumor or no tumor. Instance detection outputs a bounding box around a specific object, e.g., tumor lesion 1 and tumor lesion 2. Instance segmentation combines the two approaches and adds a pixel-wise segmentation to each instance.

Chapter 3

Literature Research

A systematic literature review (SLR) is the process to find all available publications relating to a specific research question. Goals of an SLR are to summarize the existing knowledge on a specific topic, to identify areas to improve, and to create an overview over a specific topic. The systematic approach reduces the biases during review and reveals information about varying results on the same topic [49].

This chapter is structured in the following way: In Section 3.1 the methods of how the literature research is conducted and how the research questions to be answered are introduced. Section 3.2 describes how the literature research is performed and shows the resulting literature, matching to the research question. A comparison of the matching literature is shown in Section 3.3. The results and how they impact this work are summarized and analyzed in Section 3.4.

3.1 Methodology

Systematic literature research (SLR) is the process of extracting valuable information from a tremendous amount of literature. The first step is to derive research questions from the goals of this work.

3.1.1 Research Questions

The goals of this work are described in Table 1.1, and the design goals in Table 1.2. Based on the goals we derived the three research questions (RQ), shown in Table 3.1.

TABLE 3.1: Research questions for systematic literature research

RQ	Goals	Research Question
RQ-1	DG-1–DG-5	What features do machine learning platforms for medical imaging offer?
RQ-2	DG-1	How can the machine learning models be standardized, shared and trained?
RQ-3	DG-4	How can version management for large data sets be realized?

Research question 1 is based on all goals; it is generic and could lead to insides of current machine learning platforms for medical imaging. We aim to find more details about the challenges and constraints of machine learning in the medical field and how they are currently being tackled.

Research question 2 is derived from goal *GO-1* and seeks answers about how the machine learning models can be standardized, shared and trained. We aim to find a way to standardize how machine learning models can be stored and how the whole machine learning pipeline can be designed and optimized. A standardized way of storing and using machine learning models is needed to be able to support multiple machine learning approaches within the same platform. Each machine learning approach has its own requirements and workflow. A standard aims to find a generic solution to maximize the supported machine learning approaches within our system. Having a standard also reduces the challenges to share machine learning models and use existing implementations. This research question is specific to machine learning models and does not limit on the medical context.

Research question 3 is derived from goal *GO-4* and evaluates how machine learning data sets are stored and managed. Data sets can have multiple revisions due to new images or annotations and should not be replicated after every change. To keep track of the changes and be able to keep old versions, we aim to discover the currently available approaches to apply version management on large data sets.

3.1.2 Database Search and Snowballing

To conduct an SLR two approaches are common: database searches and snowballing [44, 69, 97, 19, 50]. Database searches require a set of search queries, which are derived from the research questions and used to find related publications. The literature results are then further analyzed, filtered, grouped and reduced to a small set of publications which are closely related to the research question. It sometimes may be necessary to iteratively improve the search terms to find results which fit the resource question better. Snowballing focuses on finding related publications to a specific publication. It is divided into two parts [8]: backward snowballing (BSB) and forward snowballing (FSB). BSB focuses on analyzing the references whereas FSB focuses on the citations to find essential publications relating to that topic.

In this work, we combine search term based database search and snowballing. During the first part, a database search is performed, in the second part, the context of the best fitting publications is analyzed with snowballing. For the database search, we need to define search queries, depending on the digital database. For simplicity, only pseudo search queries are shown and accordingly adjusted for each digital library. The search queries are shown in Table 3.2. To find related publications to research question 1, we defined the search query *SQ-1.1*. We are searching for publications, where the title, abstract or the keywords are containing an exact match for *machine learning*, *medical imaging*, and *platform*. If there are more than 100 results, we analyze the first 100 most relevant matches first, and only continue if the results are promising. For the search queries *SQ-2.1* and *SQ-2.2* we included only recent publications, published after the year 2000, in order to consider only modern and currently relevant machine learning models.

After defining the research questions and search queries, we need to find matching digital databases. Therefore, we checked which databases are available for us in the scientific, medical field and chose the following five scientific, digital database libraries for our research:

TABLE 3.2: Derived search queries for the research questions

ID	RQ	Search Query
SQ-1.1	RQ 1	"machine learning" + "medical imaging" + "platform"
SQ-2.1	RQ 2	"machine learning" + "lifecycle management" + published after 2000
SQ-2.2	RQ 2	"machine learning" + "model management" + published after 2000
SQ-3.1	RQ 3	"dataset" + "version management"

- ACM[1],
- IEEE Xplore[2],
- PubMed[3],
- ScienceDirect[4],
- arXiv[5].

To evaluate which results of the search queries are relevant to our research question, we defined specific criteria, which is applied to every publication we find during database search and snowballing. The criteria can be seen in Table 3.3.

TABLE 3.3: Criteria for publications to be considered relevant

ID	Criteria
CR-1.1	Publication title is relevant to the research question
CR-1.2	Abstract is relevant to the research question
CR-1.3	Full publication is relevant to the research question

The criteria *CR-1.1* is fulfilled, when the title of the publication is close to the research question. This is the weakest criteria and easiest way to filter non-matching results. Criteria *CR-1.2* is fulfilled, when *CR-1.1* is fulfilled and the abstract is similar to the research question. When *CR-1.2* is fulfilled, the title and abstract of the publication is relevant to at least one of our research question. In that case, we evaluate the whole publication and the provided material from the authors. If the publication relates strongly with our research question it is considered highly relevant to our research. Therefore it meets the highest *CR-1.3* criteria. Additional to the database research the following publication was recommended by personal communication. Thus the publication *Applied Machine Learning at Facebook from Hazelwood et al.* [41], is added to the search results. After the database search, all publications meeting the *CR-1.3* criteria are combined as start set for the snowballing process.

The snowballing approach consists of two parts. First the backward snowballing process (BSB). During BSB references are analyzed using the criteria mentioned in Table 3.3. If the reference is matching at least *CR-1.1* it is considered relevant. To

[1]ACM Digital Library, dl.acm.org, last accessed 07.02.2019
[2]IEEE Xplore, ieeexplore.ieee.org, last accessed 07.02.2019
[3]PubMed, www.ncbi.nlm.nih.gov/pubmed, last accessed 07.02.2019
[4]ScienceDirect, www.sciencedirect.com, last accessed 07.02.2019
[5]arXiv, arxiv.org, last accessed 07.02.2019

measure how much of the references are relevant in comparison to all references within the publication, the precision is calculated:

$$P_{BSB} = \frac{related\ references}{total\ amount\ of\ references}$$

After the BSB process, the forward snowballing process (FSB) is applied. It is similar to the BSB process but instead of looking at the references of a publication, FSB evaluates which publications, which contain citations to the currently focused publication, are relevant to our research. If the reference is matching at least *CR-1.1* it is considered relevant and added to the overall publication summary. Analogous to the precision of the BSB, the precision of the FSB is calculated by:

$$P_{FSB} = \frac{related\ citations}{total\ amount\ of\ citations}$$

3.2 Literature Results

The results of the database search are shown in Table 3.4.

TABLE 3.4: Digital library search result-precision

Search Query ID	ACM	IEEE	PubMed	ScienceDirect	arXiv
SQ-1.1	0/2	4/11	1/1	1/5	0/5
SQ-2.1	0/3	2/10	0/0	0/1	0/1
SQ-2.2	1/7	0/11	0/0	0/5	0/1
SQ-3.1	1/2	0/0	1/1	0/0	2/2

Search query *SQ-1.1* for the first research question had four relevant results within the IEEE database, and one relevant result in PubMed and the same publication as well in ScienceDirect. Search query *SQ-2.1* and *SQ-2.2* answering the second research question had two relevant results within the ACM digital library and one within IEEE Xplore. For research question 3 we used search query *SQ-3.1* and found two publication in the arXiv digital library.

All publications meeting the *CR-1.3* criteria are added to the start set for the snowballing process. As described in Section 3.1, the backward and the forward snowballing process is performed. The results and amount of references and citations found for each publication, using both snowballing methods (BSB and FSB) and how much of them are relevant to a research question can be seen in Table 3.5. During backward snowballing the publications from Vartak et al. [43] had the highest precision. Within the forward snowballing process Bhardwaj et al. [14] had the best precision.

The overall summary of relevant publications, how we found them, and which criteria they matched can be seen in Table 3.6. The summary in Figure 3.1 shows only the publications matching *CR-1.3*. The publications are compared by keywords, context and motivation, goals and methods & contribution in the Table 3.7.

TABLE 3.5: Backward and forward snowballing results

Publication Title	P_{BSB}	P_{FSB}
Distributed Container-Based Evaluation Platform for Private/Large Data sets [31]	5/26	0/0
NiftyNet: a deep-learning platform for medical imaging [34]	1/57	2/49
Towards Unified Data and Life-cycle Management for Deep Learning [63]	5/39	2/22
Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective [41]	0/17	0/28
DataHub: Collaborative Data Science & Data set Version Management at Scale [14]	1/22	7/88
Goods: Organizing Google's Data sets [36]	3/26	1/53

TABLE 3.6: Systematic literature research results

Publication Title	Source	Criteria	Database	Date
Distributed Container-Based Evaluation Platform for Private/Large Data sets [31]	SQ-1.1	<=CR-1.3	IEEE Xplore	2018
NiftyNet: a deep-learning platform for medical imaging [34]	SQ-1.1	<=CR-1.3	PubMed, ScienceDirect	2018
An easy-to-use image labeling platform for automatic magnetic resonance image quality assessment [56]	SQ-1.1	<=CR-1.2	IEEE Xplore	2017
Medical Imaging Processing on a Big Data Platform Using Python: Experiences with Heterogeneous and Homogeneous Architectures [80]	SQ-1.1	CR-1.1	IEEE Xplore	2017
AggNet: Deep Learning From Crowds for Mitosis Detection in Breast Cancer Histology Images [2]	SQ-1.1	CR-1.1	IEEE Xplore	2016
GIFT-Cloud: A data sharing and collaboration platform for medical imaging research [29]	BSB	<=CR-1.3	ScienceDirect	2017
Cloud-based benchmarking of medical image analysis [37]	BSB	<=CR-1.2	SpringerOpen	2017
BEAT: An Open-Source Web-Based Open-Science Platform [6]	BSB	<=CR-1.2	arXiv	2017
Evaluation-as-a-Service: Overview and Outlook [40]	BSB	<=CR-1.2	arXiv	2015
Cloud–Based Evaluation Framework for Big Data [39]	BSB	<=CR-1.2	Springer Link	2013
Cloud deployment of high-resolution medical image analysis with TOMAAT [64]	FSB	<=CR-1.3	IEEE Xplore	2018
Bringing the Algorithms to the Data: Cloud–Based Benchmarking for Medical Image Analysis [38]	BSB	<=CR-1.2	Springer Link	2012
Towards Unified Data and Life-cycle Management for Deep Learning [63]	SQ-2.1, BSB	<=CR-1.3	IEEE Xplore	2017

table continued on next page

Table 3.6 — continued from previous page

Publication Title	Source	Criteria	Database	Date
ModelHub: Deep Learning Life-cycle Management [62]	SQ-2.1	<=CR-1.2	IEEE Xplore	2017
Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective [41]	BSB	<=CR-1.3	IEEE Xplore	2018
ModelDB: A System for Machine Learning Model Management [88]	SQ-2.2, FSB, BSB	<=CR-1.2	ACM	2016
DeepNeuro: an open-source deep learning toolbox for neuroimaging [11]	FSB	<=CR-1.3	arXiv	2018
Data Engineering: Special Issue on Machine Learning Life-cycle Management [43]	FSB	<=CR-1.2	IEEE	2018
The Missing Piece in Complex Analytics [26]	BSB	CR-1.1	arXiv	2015
DLHub: Model and Data Serving for Science [22]	FSB	<=CR-1.2	arXiv	2018
DataHub: Collaborative Data Science & Data set Version Management at Scale [14]	SQ-3.1, BSB	<=CR-1.3	arXiv	2014
Principles of Data set Versioning: Exploring the Recreation/Storage Tradeoff [15]	SQ-3.1, BSB	<=CR-1.2	PubMed	2015
Archiving Scientific Data [20]	BSB	CR-1.1	ACM	2004
Goods: Organizing Google's Data sets [36]	FSB	<=CR-1.3	ACM	2016
UStore: A Distributed Storage With Rich Semantics [28]	FSB	<=CR-1.2	arXiv	2017
DataLab: a version data management and analytics system [100]	FSB	<=CR-1.2	IEEE Xplore	2016

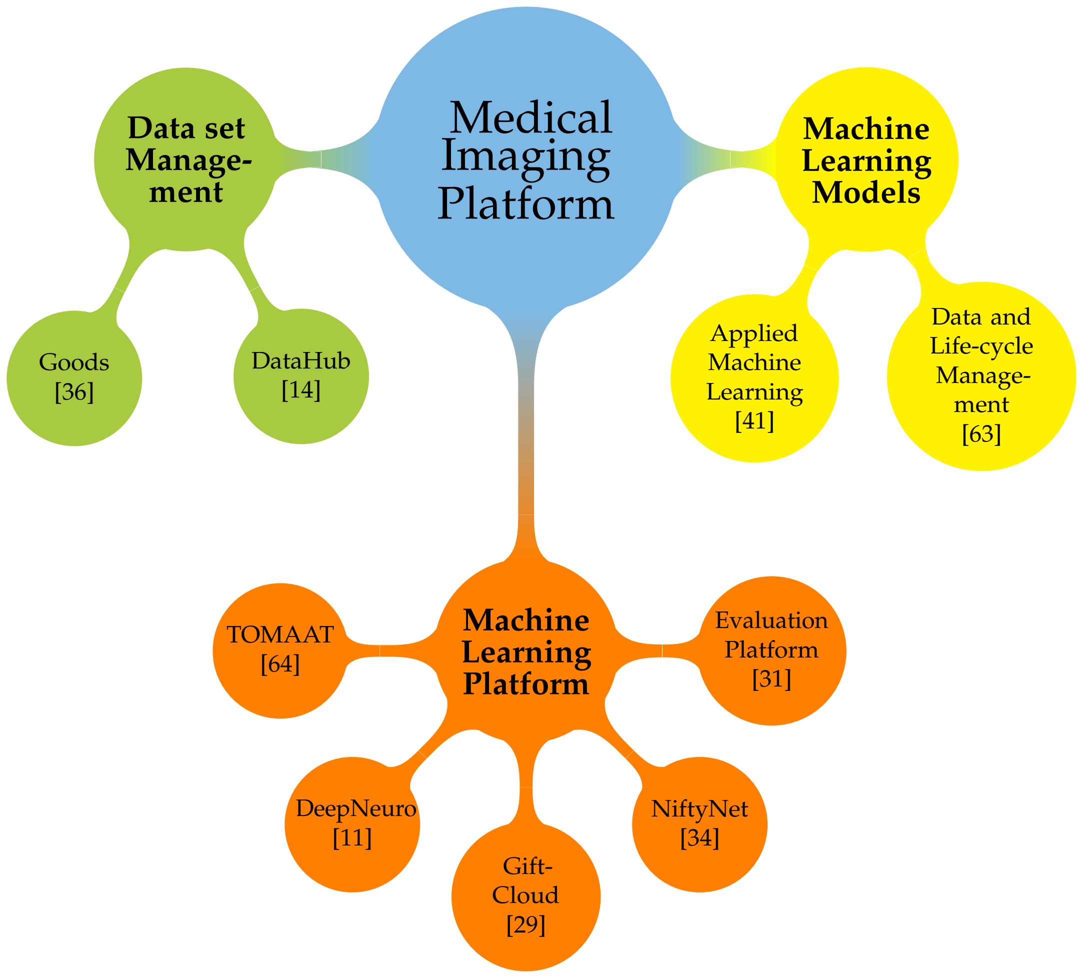


FIGURE 3.1: Systematic literature research results, summary graph

TABLE 3.7: Comparison of most relevant publications

Cite	Keywords	Context and Motivation	Goal	Method & Contribution
[31]	E-Health, Evaluation, Distributed containerization, Docker, Big data, Data privacy, Reproducibility, Containers, Computer architecture, Task analysis, Machine learning, Distributed databases, Biomedical imaging, Graphics processing units	Exponential increase in stored data, Sharing data for research is challenging, Physically moving data sets to researchers is inconvenient, Research on sensitive data is often not possible,	Novel distributed platform using containers for simple execution and evaluation of research applications on the data owner's infrastructure, bringing the algorithms to the data and avoids transfer of large data sets	Distributed container-based evaluation platform which circumvents a lot of issues in this field.
[34]	Medical image analysis, Deep learning, Convolutional neural network, Segmentation, Image regression, Generative adversarial network	Medical problems are being addressed with deep-learning-based solutions. Established deep-learning platforms do not provide specific functionality for medical context.	Open-source NiftyNet platform for deep learning in medical imaging.	Platform which allows the researchers to rapidly develop and distribute deep learning solutions for segmentation, regression, image generation, and representation learning applications, or extend the platform to new applications.
[29]	Data sharing, Biomedical research, Cross-disciplinary research, Anonymisation, Deidentification, Fetal surgery	Clinical imaging data is essential but existing systems are poorly suited for data sharing between healthcare and academia.	GIFT-Cloud, a data and medical image sharing platform, to meet the needs of GIFT-Surg, an international research collaboration that is developing novel imaging methods for fetal surgery.	Platform implemented in a multi-centre study for fetal medicine research. Case study of placental segmentation for pre-operative surgical planning, showing how GIFT-Cloud-Platform underpins the research and integrates with the clinical workflow.

table continued on next page

Table 3.7 — continued from previous page

Cite	Keywords	Context and Motivation	Goal	Method & Contribution
[64]	Deep learning, medical image analysis, segmentation, registration, cloud deployment, clinical translation	Recent research efforts have improved the state of the art, but most of the methods cannot be easily accessed, compared or used by other researchers or clinicians.	Open-source framework to provide AI-enabled medical image analysis through the network	Platform, which provides a cloud environment for general medical image analysis
[11]	Computer science, computer vision and pattern recognition, deep learning, neuroimaging, software, open-source, preprocessing, reproducibility	Translating neural networks from theory to clinical practice has unique challenges.	Best-suited deep learning framework to put deep learning algorithms for neuroimaging in practical usage.	Platform that can be used to both design and train neural network architectures, including proof.
[63]	Machine learning, training, predictive models, database management systems, inference mechanisms, artificial intelligence, storage management, data management, life-cycle management, deep learning	Current systems, focus on model building and training phases, while the issues of data management, model sharing, and life-cycle management are largely ignored.	New vision and implementation of a data and life-cycle management system for deep learning.	Extensive experiments over several real data sets from computer vision domain to show the efficiency of the proposed techniques.

table continued on next page

Table 3.7 — continued from previous page

Cite	Keywords	Context and Motivation	Goal	Method & Contribution
[41]	Facebook, neural networks, support vector machines, machine learning pipelines, high-performance distributed training flows, span machine	Describing the hardware and software infrastructure that supports machine learning at global scale, challenges in delivering data to high-performance distributed training flows, intense computational requirements	Support extremely diverse machine learning workloads, addressing emerging challenges that include machine learning algorithms, software, and hardware design	Application of machine learning to real problems at this scale, including a deep learning tools ecosystem.
[14]	Computer science, databases	Relational databases have limited support for data collaboration	A data set version control system and a platform that gives the ability to perform collaborative data analysis at scale.	Utilizing two tightly-integrated systems: A Data set Version Control System and a hosted platform on top of that.
[36]	not provided	Rethink how we organize structured data sets at scale, in a setting where teams use diverse and often idiosyncratic ways to produce the data sets and where there is no centralized system for storing and querying them	Building a large-scale enterprise-level data-management system, crawl and infer the metadata for billions of data sets, to maintain the consistency of the metadata catalog at scale	Data-Management system that is capable of extracting and exposing metadata ranging from salient information about each data set to relationships among data sets, such as similarity and provenance.

3.3 Literature Synthesis

During literature synthesis, publications are analyzed with the aim to find answers to the research questions. The research questions are described in Table 3.1. To answer the questions, multiple categories are derived from the goals and the description of this thesis and used to compare multiple approaches with each other. This categories can either be present or not present. In case no information about the attribute is available for a platform, not present is assumed.

3.3.1 Research Question 1

Research question 1 aims to analyze existing approaches for medical imaging platforms leveraging machine learning. We evaluate the following five publications:

Eval: Eggel et al., 2018, Distributed Container-Based Evaluation Platform for Private/Large Data sets, published at the 17th International Symposium on Parallel and Distributed Computing (ISPDC) [31].

NiftyNet: Gibson et al., 2018, NiftyNet: a deep-learning platform for medical imaging, published within the journal paper Computer Methods and Programs in Biomedicine [34].

GIFT: Doel et al., 2017, GIFT-Cloud: A data sharing and collaboration platform for medical imaging research, published within the journal paper Computer Methods and Programs in Biomedicine [29].

TOMAAT: Milletari et al., 2018, Cloud deployment of high-resolution medical image analysis with TOMAAT, published within the journal paper IEEE Journal of Biomedical and Health Informatics [64].

DeepNeuro: Beers et al., 2018, DeepNeuro: an open-source deep learning toolbox for neuroimaging, published on arXiv.org in the Computer Science > Computer Vision and Pattern Recognition category [11].

The comparison results based on the categories can be seen in Table 3.8. The categories are described as follows:

- **Platform Deployment Strategy:** Platforms are usually deployed using multiple strategies. The purest form of deployment is to directly install it on the machine, without further encapsulation. To use the resources more efficiently and increase the portability, containers are often leveraged. In this block, we evaluate how the platforms are deployed, especially if they are directly deployed on the operating system (using virtual machines), or if they support more advanced strategies such as container deployment. We assume in this case, that all of the platforms can either be deployed on a virtual machine or within containers. To our knowledge, the analyzed platforms did not require native hardware access without the support for containerized environments. Containerized environments are capable of running with or without underlying virtualization.

Virtual Machines: The guest operating system is encapsulated in a virtual machine (VM) on the host machine. It defines the virtual hardware the guest system is able to use (such as CPU cores, amount of memory, and network interfaces). This may increase the efficiency since virtual machines can share the resources from a host.

Containerization: Most platforms are still relying on a full operating system environment. Containerization focusses portability, scalability, and the simplification of adjusting, customizing, and replacing of components. Especially in larger container deployments, single containers can be scaled, modified, replaced, and monitored independently from the other components. Containerization methods encourage the precise definition of interfaces, communication strategies, requirements, and tasks for every component within the platform. Thus a containerized concept is similar to building the platform with small bricks, where each brick has a small task and can be replaced, shared and tested independently from the other components. This provides more freedom for the developers to create, share and adjust each brick individually as they need it.

- **Machine Learning Framework Support:** In this block, the machine learning frameworks within the model management environment are evaluated. Each framework has its strengths and weaknesses and requires specialized knowledge on its concept, implementation, and usage. Also optimizing specific needs in order to perform best within a specific situation such as optimization for debugging and development or development for production use at scale are shaping how frameworks are used in practice. The focus is on the most common frameworks Tensorflow and PyTorch.

  Tensorflow: Tensorflow [1, 5] is a popular, open-source machine learning framework developed by members from Google Brain [17]. Tensorflow supports the programming languages Python [71] and C [47]. It is also frequently used by researchers in the medical field.

  PyTorch: Similar to Tensorflow, PyTorch [75] is also a commonly used open-source machine learning framework. It is developed by Facebook and is deeply integrated into the programming language Python. PyTorch is very popular in the medical field, next to Tensorflow.

  Custom Framework: Some use-cases cannot be implemented using Tensorflow or PyTorch. In this case, the platform should be able to use other frameworks, by offering an interface to exchange the framework-component as needed.

- **Medical Customization:** Many unique challenges and specifics are present in the medical environment, which requires to be tackled accordingly. This block focuses on medical attributes that are especially needed for the medical imaging platforms.

  DICOM Support: Digital Imaging and Communications in Medicine standard [7] (DICOM) was developed and improved over many years, which introduces a significant challenge in terms of backward compatibility for a generic imaging platform. DICOM is used by most imaging related modalities in the medical field to store and transfer medical data. Therefore it is essential for a medical imaging platform to be able to handle DICOM object files and the DICOM communication protocol.

NIFTI Support: Neuroimaging Informatics Technology Initiative [65] (NIFTI) has a medical imaging data format. In contrast to DICOM, it is frequently used for analysis, designed to support scientific visualization and image processing, and especially the metadata is not standardized.

NRRD Support: Nearly Raw Raster Data [83] (NRRD) is also a medical imaging data format, similar to NIFTI.

PACS Interface: The Picture Archiving and Communication System [12] (PACS) is a system used in the clinical environment, which provides services for the storage and access to medical images relying on the DICOM image format.

Anonymisation: Anonymisation is very crucial to ensure sensitive patient information are never leaked to the outside world. Some platforms introduce specialized gateways to access sensitive patient information but ensure that all of the personal information is removed or replaced prior providing it to the user.

Custom Processing Pipelines: Processing pipelines are the backbone of the machine learning platform. They combine all tasks needed to perform machine learning tasks into one schedule. Customization capability is essential to adjust certain tasks to the specific needs to a new use-case.

Table 3.8 compares the most relevant publications to research question 1 with each other. Two of the five platforms are based on modern containerization approaches [31, 11], the others are relying on virtual machines for deployment.

The essential component for machine learning is the framework and programming language used to perform the training and inference. In this case, most platforms support the machine learning framework Tensorflow and PyTorch. Three of the compared platforms allow exchanging the used machine learning framework for a custom one and therefore support multiple machine learning frameworks [31, 11, 64].

Another crucial factor is the customization for the medical field. In the medical field, many additional challenges occur, which are very complicated to tackle. One challenge is how the images and their metadata are stored within a clinical environment. Fortunately, all platforms except [31] provide native DICOM Object file support. A frequently used workaround is to transform DICOM objects to other file formats and work with these formats. Alternative formats such as NIFTI and NRRD are mostly used during scientific research. Three platforms provide native support for NIFTI images [34, 29, 11] and only one supports the NRRD file format [11]. Another challenge is how the images are accessed and used for scientific research. Clinics usually store the patient images within the Picture Archiving and Communication System. It relies on the DICOM network protocol to access and store patient data and thus has to comply with high-security standards. During research we found only one

TABLE 3.8: Literature synthesis: Research question 1

Category	Eval. [31]	NiftyNet [34]	GIFT [29]	TOMAAT [64]	DeepNeuro [11]
Deployment:					
Virtual Machines	no	yes	yes	yes	no
Containerization	yes	no	no	no	yes
Framework Support:					
Tensorflow	yes	yes	no	yes	yes
PyTorch	yes	no	no	yes	yes
Custom Framework	yes	no	no	yes	yes
Medical Customization:					
DICOM Support	no	yes	yes	yes	yes
NIFTI Support	no	yes	yes	no	yes
NRRD Support	no	no	no	no	yes
PACS Interface	no	no	yes	no	no
Anonymisation	no	no	yes	no	no
Custom Pipelines	no	no	no	no	yes

platform tackling this critical problem [29]. It offers specialized gateways as intermediate, which is the only connection to the clinical PACS. Thus it can be ensured that all the critical patient data can be anonymized prior to scientific usage. The same platform is also the only one including features for anonymization of patient data and reducing the barriers connecting the researchers with real data. To apply machine learning models to the anonymized patient data, custom processing pipelines are frequently needed to adjust each processing step to the specific task. In this case, only one publication implemented a more dynamic approach, which provides the user with the ability to customize the whole machine learning processing pipeline [11]. In contrast to the customized framework, this also includes tasks such as importing medical patient images and exporting processed images to the clinical PACS.

These platforms offer a variety of essential features, but none of them implemented all important features. GIFT [29] was the only publication to tackle the management of sensitive patient data but lacks support for common machine learning frameworks. DeepNeuro [11] provides many of features but lacks PACS and anonymization support.

3.3.2 Research Question 2

Research question 2 aims to analyze existing approaches to standardize, share and train machine learning models within a machine learning platform, independently from the medical context. We are evaluating the two following publications:

Data Management: Miao et al., 2017, Towards Unified Data and Life-cycle Management for Deep Learning, published within the IEEE 33rd International Conference on Data Engineering (ICDE) [63].

ONNX: Hazelwood et al., 2018, Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective, published within the IEEE International Symposium on High Performance Computer Architecture (HPCA), especially the Open Neural Network Exchange (ONNX) format [21].

The comparison results based on the categories can be seen in Table 3.9. For comparison the following categories used:

- **Generic Attributes:** Working with machine learning models requires lots of different tools and skills. Combining these tools and providing more possibilities and guidance within the platform can increase the development efficiency for researchers.

  Transfer Model between Frameworks: Many machine learning frameworks are available for various reasons and optimized for different tasks and hardware. The first step towards standardization is the capability of transferring models between frameworks (decoupling models from specific frameworks). Ideally, it would be possible to develop a machine learning model in a framework which suits best for development and later transforms it into another framework which is the best for production usage on the target hardware platform.

  Model Standardization: A standardized description of a model makes it independent from the framework. In the ideal case, a platform supporting the standard model representation would remove the dependency on a specific machine learning framework.

  Model Life-cycle Tracking: Machine learning models are usually developed, trained, applied and improved iteratively. Life-cycle tracking monitors and documents each iteration step to describe how each model was trained. This helps the developers to visualize the process and the history to further push the limits for the next iteration.

  Model Zoo: A model zoo is a collection of pre-trained models for specific tasks. It represents a shared, central repository for models within the platform. Users can access the available models and use them as a reference for developing new models, to compare the performance of new models with existing ones or other research purposes.

  Model Search Query: Model queries offer the possibility to search a model zoo by looking at the metadata and other attributes of the available models.

- **Machine Learning Framework Support:** In this block, the machine learning frameworks within the model management environment are evaluated. Each framework has its strengths and weaknesses and requires specialized knowledge on its concept, implementation, and usage. Also optimizing specific needs in order to perform best within a specific situation such as optimization for debugging and development or development for production use at scale are shaping how frameworks are used in practice. The focus is on the most common frameworks Tensorflow and PyTorch.

- **Tensorflow:** Tensorflow [1, 5] is a popular, open-source machine learning framework developed by members from Google Brain [17]. Tensorflow supports the programming languages Python [71] and C [47]. It is also frequently used by researchers in the medical field.
- **PyTorch:** Similar to Tensorflow, PyTorch [75] is also a commonly used open-source machine learning framework. It is developed by Facebook and is deeply integrated into the programming language Python. PyTorch is very popular in the medical field, next to Tensorflow.
- **Other Frameworks:** Some use-cases cannot be implemented using Tensorflow or PyTorch. In this case, it should be possible to use other frameworks within the same model management environment.

TABLE 3.9: Literature synthesis: Research question 2

Category	Data Management [63]	ONNX [41, 21]
Generic Attributes:		
Transfer Frameworks	no	yes
Standardization	no	yes
Life-cycle Tracking	yes	no
Model Zoo	yes	yes
Model Query	yes	no
Framework Support:		
Tensorflow	yes	yes
PyTorch	yes	yes
Other Frameworks	yes	yes

Table 3.9 compares the most relevant publications to research question 2 with each other. There are many different machine learning frameworks available, which are defining how the model is defined and executed. Ideally, it is possible to transform one model, independently from the framework it was developed in, into another framework and its architecture. The capability of importing a model from a framework and exporting it to a model of another framework is offered by one publication [41]. They achive this by adding the feature of transforming the model into an intermediate standardized representation, namely ONNX. From this intermediate representation, they are able to export the model to another framework. This approach is very promising since it aims to build a well-defined interface between frameworks to allow the developers to focus on the model itself instead of the framework which is used to execute it later. Unfortunately, they do not provide life-cycle tracking for the development of the model, for the developers at this time. This was implemented by [63], and shows the development history, including the lineage. Another important factor is the availability of a reference model repository (Model Zoo). It provides the possibility to share pre-trained models with others, and have a local collection of multiple models for a specific set of tasks. This is implemented by both publications [62, 41]. Additionally to the model zoo, it is beneficial to design

specific search queries in order to find the right model for a specific set of tasks. This is implemented by [63].

Both publications support Tensorflow, PyTorch and other machine learning frameworks.

3.3.3 Research Question 3

Research question 3 aims to analyze existing approaches to store and manage data sets used for machine learning, including version management. We evaluate the following two publications:

DataHub: Bhardwaj et al., 2014, DataHub: Collaborative Data Science & Data set Version Management at Scale, published at arXiv [14].

Goods: Halevy et al., 2016, Goods: Organizing Google's Data sets, published within the conference paper Proceedings of the 2016 International Conference on Management of Data [36].

The comparison results based on the categories can be seen in Table 3.8. The categories are described as follows:

- **Generic Attributes:** Managing data sets for machine learning is a very challenging and underestimated task. Therefore, the following attributes are compared.

  API-Access: Data sets are usually used within processing pipelines during training. These are highly automated and executed without user interaction. Therefore it is essential to provide an interface to connect the database management system with these pipelines.

  Custom Data Formats: Depending on the origin of the data, it can have very different attributes and shapes. Data sets can consist of objects such as large, high-dimensional images, and comma-separated values or videos. Therefore the data set management system should allow storing different types of data.

  Change Protection: Data sets are used in conjunction with machine learning. To reproduce the machine learning tasks, the exact, unchanged version of the data set is required. Therefore the data set management system should prevent changes to the data set, but allow the creation of new data sets with the same data.

  Branching: Branching allows data sets to change, without replicating all the data within the data set. This can be implemented by linking to the existing data set and only store the changed data.

  Lineage Information: Data sets often evolve over a long time. To evaluate the process and identify problems or positive parts of the history lineage information is essential.

  Search Query: Having a large number of data sets available, it can be tough to find data sets matching specific criteria. Therefore the database management system should provide the capability to execute search queries and provide related data sets.

TABLE 3.10: Literature synthesis: Research question 3

Category	DataHub [14]	Goods [36]
Generic Attributes:		
API-Access	yes	yes
Custom Data Formats	yes	yes
Change Protection	yes	yes
Branching	yes	yes
Lineage	yes	yes
Search Query	yes	yes

Table 3.10 compares the most relevant publications to research question 3 with each other. In this case, both publications were able to support all our generic attributes. They both offered API-Access to allow other programs to access the data sets directly, allow the storage of custom data, protect the data set from changes, allow branching of data sets, provide lineage information on how the data sets evolved and accept search queries to find relevant data sets based on the search query.

3.4 Takeaway

During literature research, we defined three research questions to define the focus of our research. The research questions are shown in Table 3.1.

Research question 1 evaluates how current medical imaging platforms are deployed, which frameworks they support and which medical customizations were applied to establish them in the medical field, as described in Section 3.3.1. The approach taken by DeepNeuro [11] is very promising, leveraging containerization to exchange parts of the machine learning pipelines similar to bricks, and allowing the researchers to customize the bricks as required. With this approach, the interfaces to each brick must be clearly stated to ensure that new bricks are matching those before they are integrated with the platform. Unfortunately, they do not provide quality assurance methods, which test each component before they are accepted as part of the platform. Another important aspect is the integration into the medical environment. The DeepNeuro platform offers support for multiple file formats, such as DICOM, NIFTI, and NRRD, but we were not able to find the capability to connect to common medical image management systems such as PACS or XNat. Additionally, they do not have any precautions or tools to ensure that sensitive patient data is removed (anonymized) or only accessible in a specific, restricted way. This was well tackled by the GIFT-Cloud platform [29], which added many additional features and components to ensure all sensitive data is handled accordingly and anonymized if needed.

Research question 2 takes a broader look at standardization, generic features and life-cycle management of machine learning tasks, and how existing models can be used, even if they are written within unsupported frameworks by transferring them into a support framework or own representation of the model, as described in Section 3.3.2. The publication Data and Life-cycle Management [63] developed important approaches to tackle the machine learning pipeline, from collecting the data and machine learning models, training, evaluating and serving the newly trained model for inference. It is crucial to understand how new machine learning models are developed to adjust the platform accordingly. Additionally, they developed a model registry to collect the pre-trained models and share them with other researchers.

Research question 3 summarizes existing approaches to manage data sets at scale, including version control, branching, custom data formats and strategies to efficiently find relating data sets within a large data set repository. The publications Goods [36] and DataHub [14] are very promising. They are capable of handling a vast amount of data types and sizes, analyze the data sets autonomously to extract relevant metadata, track how they evolve over time and create new data sets by branching as needed.

It would be very promising to design and implement such a concept, however this would require significant efforts of multiple developers due to the high amount of functionality and the high complexity. Therefore, we are planning to create a concept of the component based foundation of a platform that fit our needs and extend the functionality iteratively in future work. We are defining the requirements for our platform in Chapter 4.

Chapter 4

Requirements Analysis

This chapter describes the necessary requirements for the Medical Machine Learning Platform (MMLP), following the task and object-oriented requirements engineering (TORE) approach [72].

During the elicitation of requirements, the insights of the literature research in Chapter 3, and in particular, the work by Rupp et al. [78], Nuseibeh et al. [70], and Kotonya et al. [51] are used.

Section 4.1 explains how requirements for the platform were derived from the insights about the medical domain and machine learning. Section 4.2 describes the domain data, especially the application of machine learning in the medical field. The primary personas are identified and described in Section 4.3. By analyzing the motivation, interests, and background of each persona, requirements of the platform are derived. With these personas in mind, we take a more in-depth look at the data that these personas work with and introduce the required terminology (used throughout this thesis). To further describe the user interaction and the platform behavior, the functional requirements are derived in Section 4.4. Subsequently, the interaction between the user and the platform is described, focusing on the complete workflow the user has to perform to fulfill a specific task in Section 4.5. Collaborative workflows are introduced in Section 4.6. These are followed by the non-functional requirements in Section 4.7, which describe the quality requirements for the platform. The accumulated knowledge of the previous sections is used to create the user interface structure in Section 4.8. All requirements are summarized in Section 4.9. The requirements are specified, documented, and made available to design the resulting platform, which is described in Chapter 5.

In this chapter, we used the following color-code for figures: objects related to data set management, machine learning model management, machine learning method management, and computation environment management are indicated by blue, red, green, and yellow, respectively.

4.1 Requirements Elicitation

To model the requirements according to the users' needs, multiple steps were performed. In the first step, the overall goals were defined in Chapter 1. These overall goals were used to derive the research questions answered by the systematic literature research. The literature research (see Chapter 3) yielded a broad overview of topics such as the problems, approaches, and further details about the medical context.

To further focus on the actual users of the platform, fifteen potential users of the platform were observed while working and interviewed afterwards. Contextual and one-on-one interviews yielded essential information about their work environment, such as how they actually work on a daily basis and potential areas to focus on. During interviews, mockups, schematics, and prototypes were shown to the potential users of the platform and discussed, to elicit requirements or to refine the elicited requirements in the literature research and additionally collect ideas about how the platform could be designed and operate.

Multiple users had concerns to participate, due to confidential workflows that they perform or confidential data that they work with. Further, it seemed that their current workflow was not mature enough that they wanted it to be shared or published. Therefore, no personal or work-topic related information was collected, only the fundamental approaches required to ensure the proposed platform does support, and possibly exceeds their expectations.

4.2 Domain Data Description

In this section, the terminology is introduced, focusing on the context of machine learning in the medical field.

The domain data model in Figure 4.1 describes the relationships between data sets, algorithms, parameters, hyperparameters, models, patient data, and methods: An *algorithm* describes a programmatic way of archiving a specific goal [67]. It defines all inputs, a step-by-step procedure, which describes how the input is processed and the corresponding results, and the output of the pipeline. The source code covers all the files, which are needed to run the algorithm, excluding the input data. Machine Learning algorithms optimize internal weights, or *parameters*, to minimize a user-defined loss function, such as the mean squared error. For example, parameters could be activation weights and biases in neural networks or decision rules in decision trees. Variables regarding the configuration of the algorithm, are called *hyperparameters*. In neural networks, typical hyperparameters are, for example, the optimizers learning rate, the number of epochs, and data augmentation configurations. The combination of algorithm, parameters, and hyperparameters is called *machine learning model* or short model. Thus, a model describes how the computer can learn — by minimizing the loss function — to solve a specific task [67]. The minimization of the loss function is usually done iteratively utilizing previously annotated input data. This process is called *training*.

Similar input data to train the model is grouped and stored in *data sets*. If the data set is used for training, it is also referred to as *training data*. The data sets can have very different characteristics, such as the dimensionality and format. As an example, patient images with annotations, non-medical data collections, or entirely fictional toy data sets, could be used to train the model.

Before the data set is used, it has to be *pre-processed*, which transforms the data into a for the model matching format. If the model created new images or annotations, it could be necessary to transform the data back to the original format. This step is called *post-processing*. These data processing steps vary depending on the intention of the user. If the user wants to train the model, the pre-processing part could be applied

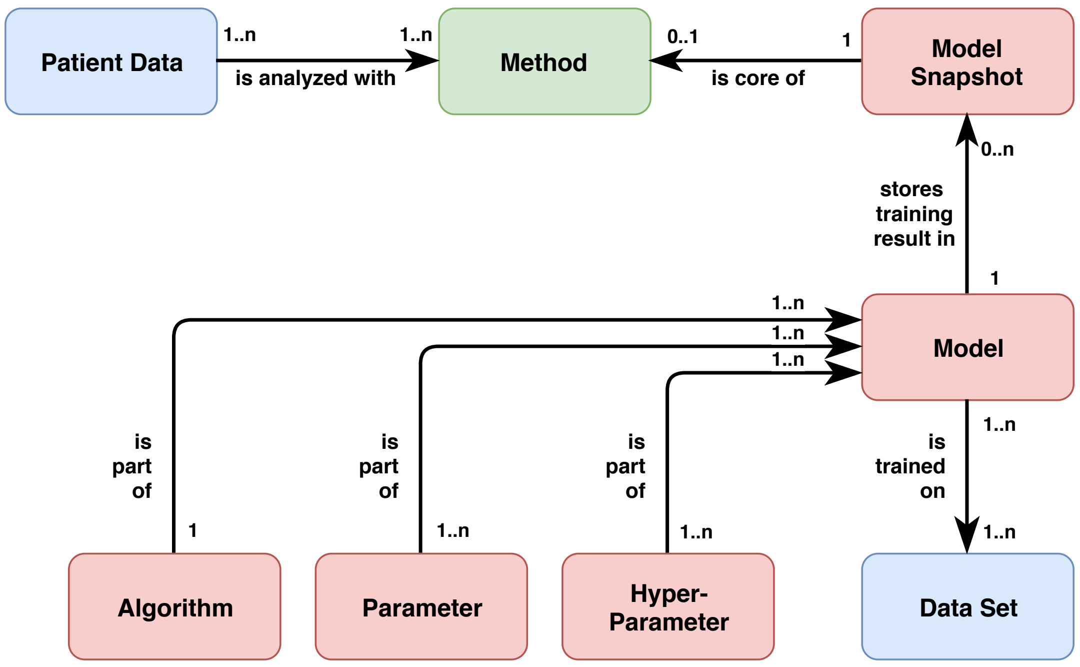


FIGURE 4.1: Domain data diagram: Visualizing the relationships between data sets, algorithms, parameters, hyperparameters, models, patient data, and methods.

manually before uploading the data set. This reduces the training time because the data set does not need to be transformed prior to every model training iteration.

After the training is completed, the model and its parameters are frozen and stored separately. The combination of the model and the learned parameters is called *model snapshot*. This model snapshot can now be used to create a *machine learning method* or short method. A method is able to use the model snapshot on previously unseen, but similar input data, to generate annotations, which is called *inference*. In addition to the model snapshots, all configuration and required steps, e.g., pre- and post-processing are automated. This helps medical experts to try available machine learning methods on the patient data they are working with. Medical experts work with various types of data they collect from their patients. For segmentation tasks, medical experts are using patient cohorts, which are patient images grouped by patient attributes, e.g., all patients between the age of 42 and 47, female, and specific types of illnesses.

Training and inference of machine learning models and methods need a lot of computational resources. To tackle these demands, custom computing environments are leveraged.

4.3 Primary Personas

The platform should be designed as closely as possible to the actual user needs. Therefore, the first consideration is what kind of users should benefit from the platform. To describe the users, a persona is utilized, which describes a hypothetical archetype of an actual user [24]. The proposed platform should be used by both medical experts and clinical data scientists. In general, medical experts are focused on medical images analysis, whereas clinical data scientists are focused on the development of machine learning models. The clinical data scientist can leverage the platform to create a fully automatic method, including pre- and post-processing, and share the resulting methods with other users of the platform, particularly medical experts. Medical experts can then conveniently apply the shared method to their patient data without any manual intervention. They are described in Table 4.1 and Table 4.2.

TABLE 4.1: Persona: Medical expert

Topic	Description
Biographic Facts	Has a medical background and is working with patient data, including images. Due to a large number of patient data, she wants to leverage machine learning methods to support her workflows. Typical workflows are automatically segmenting tumors in MRI scans or classifying electrocardiograms, finally resulting in a diagnosis, risk evaluation, or treatment procedure.
Knowledge	Expert level knowledge about medical-related questions, an only basic understanding of machine learning concepts and computer science, knows that it is possible to leverage machine learning techniques to analyze patient data.
Needs	Since she does not have a customized computer for machine learning and works from multiple locations, she wants to apply the tools via an intuitive, easy to use interface. Furthermore, she looks for options to apply machine learning methods on patient data of interest, and subsequently access the generated outcome.
Frustrations	Providing the patient data and applying the required machine learning methods are complicated, time-consuming, and not intuitive.
Ideal Features	Developed methods should support the medical expert in delivering superior patient care with reduced manual effort. The medical expert strongly prefers a simple and easy-to-use interface.

TABLE 4.2: Persona: Clinical data scientist

Topic	Description
Biographic Facts	Has a scientific background and is currently developing new machine learning models. She wants to bring the state-of-the-art for machine learning into the medical field and help medical experts diagnosing patient data. Therefore she improves the currently available machine learning models in the medical domain. She uses the platform to manage, develop, and share machine learning models and developed machine learning methods.
Knowledge	Expert level knowledge about machine learning and related questions, a basic understanding of the medical context. The clinical data scientist is aware of current challenges in the medical field and develops state-of-the-art machine learning methods to tackle these.
Needs	The clinical data scientist needs a platform to deploy her methods into clinical practice. Thus, she needs a simple way to access a machine learning platform remotely using a web-interface, add, customize, modify, and train machine learning models. After the models are trained, they are used to create and share new methods with medical experts in the clinical environment. Methods in the platform must include all pre- and post-processing steps, parameters, and settings needed to apply them to the patient data they are working with.
Frustrations	Platform usage is too complicated or time-consuming and restricts her flexibility to design machine learning models. She also fears a bad reputation if these machine learning models do not work as intended.
Ideal Features	Increase the popularity of machine learning models in the medical field. Improve patient care by providing better technological assistance for the medical experts.

4.4 Functional Requirements

This section describes the functional requirements of our platform. The functional requirements specify functionality that the platform must be able to perform [76]. The following sections describe what the platform needs to do, starting with the use cases in Section 4.4.2, the user-tasks in Table 4.3, and the platform functions and user stories in Section 4.4.3.

4.4.1 User Tasks

A user-task is used to model the work a persona needs to do. Additionally, sub-tasks, system-functions, and user story are used to clarify the needs further. The user-tasks are defined in Table 4.3.

TABLE 4.3: Description of user-tasks

User-Task	Description
UT-Train	Train Machine Learning Model: **Description:** Clinical data scientists need to train machine learning models. **Frequency:** The user trains a model multiple times per day on average. **Trigger:** The user starts the training whenever she likes. **Risk:** In case a data set or a model would not be supported by the platform, the user would need to develop them without using the platform, which would be a big problem for the acceptance rate and possibly cause the frustrations dealing with restrictions. **Solution Conditions:** Developing new machine learning models requires management of the model source code and its versions, data sets to train the model, and configuring and monitoring the training pipeline. Data sets and models can vary heavily depending on the type, size, structure, and content.
UT-Share	Export Machine Learning Model as Method: **Description:** The clinical data scientist needs to share the trained models (snapshots) with other data scientists and medical experts, who may want to test other models for their problem. **Frequency:** The user shares one model every three days on average. **Trigger:** The user shares a model whenever she likes. **Risk:** The user has issues sharing the model and exposing the created method. **Solution Conditions:** To share the model snapshots, the user preconfigures the pre- post- and inference-pipeline and combine it into a method. The user uses the methods to analyze or test the performance of multiple methods her patient data. Therefore various types of data and methods have to be supported.
UT-Analyze	Analyze Patient Data leveraging Machine Learning Method: **Description:** Medical experts apply machine learning methods to analyze patient data and provide important insights, such as detecting areas of interest. **Frequency:** The user uploads and analyzes data multiple times per day on average. **Trigger:** The user starts analyzing her data whenever she likes. **Risk:** The user has issues uploading and analyzing her data, due to restrictions of the platform. Additionally, the user might have issues understanding how to use the platform.

table continued on next page

Table 4.3 — continued from previous page

User-Task	Description
	Solution Conditions: The user needs to be guided to a simplified workflow to upload the patient data of interest, select the desired method, and start the analysis pipeline.

4.4.2 User Sub-Tasks

A sub-task is used to further specify the user-task. In this thesis, we use two levels of subtasks. The first level is called sub-tasks categories which are shown and described in Table 4.4. The categories are further refined in Table 4.5, Table 4.7, and Table 4.8.

TABLE 4.4: Description of sub-tasks categories

Sub-Task Category	Description
ST-DS	Manage Data Sets: **Description:** The clinical data scientist needs to manage data sets to train machine learning models. For example, they need to upload and view them. **Solution Conditions:** The data set needs to match the requirements of a machine learning model as good as possible to achieve the best possible results during training of a machine learning model. Therefore the platform has to manage the data set itself as well as supporting the user with the decision, whether a data set matches her needs or not (e.g., by showing her detailed information). **Supports user-task:** UT-Train
ST-MO	Manage Machine Learning Models: **Description:** The clinical data scientist needs to manage machine learning models prior training. For example, they need to import and view them. **Solution Conditions:** Machine learning models are an attempt to improve the number of tasks a machine can perform. The models itself are made available and are managed by the platform. In addition to the metadata about the model itself, information about tasks such as configuration, monitoring and pre-processing has to be communicated and adjusted by the user. The main purpose of models is to be improved by training and then used to create automated machine learning methods. **Supports user-task:** UT-Train, UT-Share
ST-ME	Manage Machine Learning Methods:

table continued on next page

Table 4.4 — continued from previous page

Sub-Task Category	Description
	Description: The medical expert and the clinical data scientist need to manage machine learning methods before using them to analyze patient data. For example, they need to view them and be able to check the provided description. **Solution Conditions:** Machine Learning Methods are tools that are capable of performing pre-trained tasks. The method itself is based on a model snapshot, which was trained for a specific task. In comparison to models, the method is pre-configured. It does not require any input from the user, except the patient data it should analyze. **Supports user-task:** UT-Share, UT-Analyze
ST-CE	Manage Computation Environment: **Description:** The medical expert and the clinical data scientist need to manage the compute environment. For example, they need to set up and start the training/analysis pipeline and and monitor the current status. **Solution Conditions:** The computation environment executes the training of machine learning models and inference of machine learning methods. It covers all necessary steps from the configuration, deployment, monitoring, and results management. Usually, models require a lot of customization, depending on the selected data set. During training, the user tries to improve the performance of a model, and during inference, a model is applied to the uploaded patient data. **Supports user-task:** UT-Train, UT-Analyze

TABLE 4.5: Sub-Tasks: Data set management

Sub-Task	Description
ST-DS-1	View Data Sets: **Description:** The user views all data sets, their versions, and also read the metadata to prepare the training of a machine learning model. **Persona:** Clinical Data Scientist
ST-DS-2	Upload Data Set: **Description:** The user needs to upload data sets, that matches the requirements of a specific machine learning model. **Persona:** Clinical Data Scientist

table continued on next page

Table 4.5 — continued from previous page

Sub-Task	Description
	Conditions: If the user provides additional metadata in the data set, it should be recognized and added to the description of the data set. The data set and the metadata must be valid and readable by the platform.
ST-DS-3	Add new Data Set Version: **Description:** The user needs to add a new version of the data set and keep track of the previous versions, after she adjusted the data set. **Persona:** Clinical Data Scientist **Conditions:** Previous data set versions are crucial to reproduce training results. Therefore, the user needs to keep track of differences between data sets and precise versioning of data sets and their versions. The old version of the data set has to stay available. The data set receives an incremental update, including a new unique identifier for the updated version.
ST-DS-4	Edit Data Set Metadata: **Description:** The user needs the ability to improve the documentation and the metadata of a data set. **Persona:** Clinical Data Scientist **Conditions:** The metadata includes information such as the source, maintainer, description, included patient ids and other essential characteristics, and the dates of each version. Before the metadata is modified, the platform shall verify, that the changes are valid, if possible (medical attributes are difficult to be validated autonomously).
ST-DS-5	Remove Data Sets: **Description:** The user needs to remove data sets which are no longer required. **Persona:** Clinical Data Scientist **Conditions:** The platform should only remove data sets which are not in use. Additionally to the deleted data set, all versions should be recursively removed.

TABLE 4.6: Sub-Tasks: Model management

Sub-Task	Description
ST-MO-1	View Machine Learning Models: **Description:** The user should be able to view all machine learning models and also read the metadata to check the details and available versions. **Persona:** Clinical Data Scientist
ST-MO-2	Import Machine Learning Model:

table continued on next page

Table 4.6 — continued from previous page

Sub-Task	Description
	Description: The user needs to import machine learning models, stored in external repositories. **Persona:** Clinical Data Scientist **Conditions:** If the user provides additional metadata within the model, it should be recognized and added to the description. All available versions of the model have to be recognized and added to the model-version list, including the available metadata for each version.
ST-MO-3	Update Machine Learning Model: **Description:** The user needs to tell the platform to update the source code of the machine learning model from an external repository. **Persona:** Clinical Data Scientist **Conditions:** The user notifies the platform, that it should check and import new model versions, including the metadata and change-description of each version. The old version of the machine learning model has to stay available, to repeat a training (run a training process using the same data set version, model version and hyperparameter settings as before). The machine learning model receives an incremental update, including a new unique identifier for the updated version.
ST-MO-4	Edit Machine Learning Model Metadata: **Description:** The user needs the ability to modify the documentation and the metadata of a machine learning model. The metadata includes information such as the source-url, maintainer, description, model parameters, monitoring possibilities, and the dates of each version. **Persona:** Clinical Data Scientist **Conditions:** Before the metadata is modified, the platform shall verify, that the changes are valid, if possible (medical attributes are difficult to be validated autonomously).
ST-MO-5	Remove Machine Learning Models: **Description:** The user needs to remove models which are no longer required. **Persona:** Clinical Data Scientist **Conditions:** The platform should only remove model which are not in use. Additionally to the deleted model, all versions should be recursively removed.
ST-MO-6	View Snapshots of a Machine Learning Model: **Description:** The user needs to view the available model snapshots. She requires two views: one overview of all available model snapshots, and an overview of model snapshots for a specific model version. Additionally, the user needs to see the current status of the training (status messages such as pre-processing, training, etc.)

table continued on next page

Table 4.6 — continued from previous page

Sub-Task	Description
	Persona: Clinical Data Scientist
ST-MO-7	Remove a Snapshots of a Machine Learning Model: **Description:** The user should needs to remove a specific snapshot of a model. **Persona:** Clinical Data Scientist **Conditions:** The platform should only remove model snapshots which are not in use.

TABLE 4.7: Sub-Tasks: Method management

Sub-Task	Description
ST-ME-1	View Machine Learning Methods: **Description:** The user needs to view all available machine learning methods and also read the metadata to prepare the analysis of patient data. **Persona:** Medical Expert, Clinical Data Scientist
ST-ME-2	Create Machine Learning Method: **Description:** The user needs to create a new machine learning methods. The user provides all required settings to automate the usage of the method, such as pre- and post-processing pipelines, a machine learning model, its hyperparameters, and a pre-trained snapshot. **Persona:** Clinical Data Scientist
ST-ME-3	Remove Machine Learning Method: **Description:** The user needs to remove a specific machine learning method. **Persona:** Clinical Data Scientist **Conditions:** The platform should only remove methods which are not in use.
ST-ME-4	Upload Patient Data: **Description:** The user needs to upload locally available patient data to the platform for further analysis. **Persona:** Medical Expert, Clinical Data Scientist

TABLE 4.8: Sub-Tasks: Computation environment management

Sub-Task	Description
ST-CE-1	Subscribe to Training and Inference Updates:

table continued on next page

Table 4.8 — continued from previous page

Sub-Task	Description
	Description: The user requires information about the status of the training/inference to continue with their workflow. The user needs to place a subscription, that notifies her about essential updates of a machine learning pipeline. **Persona:** Medical Expert, Clinical Data Scientist **Conditions:** Some training or inference processes will need a long time to finish. Therefore the user should be able to receive a notification outside of the platform, such as SMS, a chat-bot, or email, in case the training or inference process reaches a milestone or finishes. This notification should also include the result (successful / failed) and the logs collected during runtime.
ST-CE-2	Start Automated Inference using a Machine Learning Method: **Description:** The user needs to deploy an automated inference process, on previously uploaded patient data. **Persona:** Medical Expert, Clinical Data Scientist
ST-CE-3	Monitor and Terminate Machine Learning Pipelines: **Description:** The user needs to monitor the progress of multiple machine learning pipelines simultaneously and terminate them if she detects a problem. **Persona:** Medical Expert, Clinical Data Scientist
ST-CE-4	View and Download Results: **Description:** The user needs to access and download the results of a finished analysis of a machine learning method. **Persona:** Medical Expert, Clinical Data Scientist **Conditions:** The results should also contain comprehensive information about the individual phases during application of a machine learning method (e.g., annotations, statistics, and logs). Due to the highly varying architectures of machine learning models, the platform needs to provide a generic way of storing and offering the results to the users. Additionally, it should be easily visible who applied the method and whether it succeeded or not.
ST-CE-5	Training of a Machine Learning Model: **Description:** The user needs to configure and launch a training pipeline in the platform. **Persona:** Clinical Data Scientist **Conditions:** The training details should be documented and included in the model snapshot.
ST-CE-6	Track a single Machine Learning Training and Inference Progress:

table continued on next page

Table 4.8 — continued from previous page

Sub-Task	Description
	Description: The user should be able to track the process of a specific machine learning pipeline. **Persona:** Clinical Data Scientist
ST-CE-7	Customize the Pre- and Post-Processing Pipeline of a Machine Learning Model: **Description:** The user needs to adjust the pre- and post- pipelines for the training of a machine learning method. **Persona:** Clinical Data Scientist
ST-CE-8	Initialize Machine Learning Model using a Snapshot: **Description:** The user needs to use a previously created model snapshot to initialize a new training pipeline. **Persona:** Clinical Data Scientist **Conditions:** The model snapshots should be stored and visualized in a way that allows the user to understand how a specific snapshot was created easily, e.g., show the utilized data set.
ST-CE-9	Customize the Hyperparameters of a Machine Learning Model: **Description:** The user needs to configure the hyperparameters of a machine learning model before starting the training pipeline. **Persona:** Clinical Data Scientist

A summary of all use cases is shown in Figure 4.2. The clinical data scientist uses all functionalities of the platform during the development of machine learning models, as well as creating and testing machine learning methods. On the other hand, the medical expert should strictly focus on the application of machine learning methods on the uploaded patient data. Specifically, her goal is not to modify any methods nor fine-tune existing models to create methods.

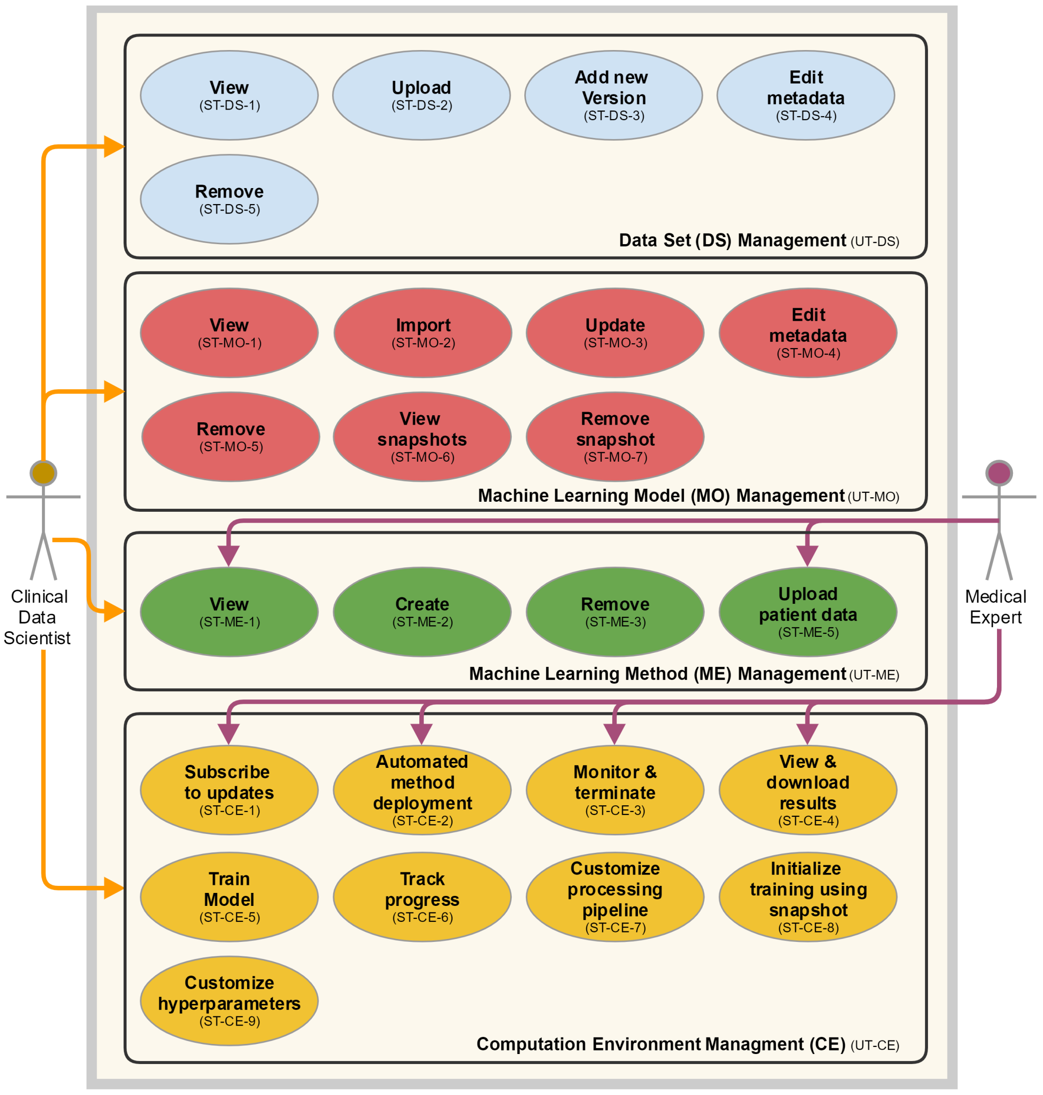

View (ST-DS-1)
Upload (ST-DS-2)
Add new Version (ST-DS-3)
Edit metadata (ST-DS-4)
Remove (ST-DS-5)
Data Set (DS) Management (UT-DS)
View (ST-MO-1)
Import (ST-MO-2)
Update (ST-MO-3)
Edit metadata (ST-MO-4)
Remove (ST-MO-5)
View snapshots (ST-MO-6)
Remove snapshot (ST-MO-7)
Machine Learning Model (MO) Management (UT-MO)
View (ST-ME-1)
Create (ST-ME-2)
Remove (ST-ME-3)
Upload patient data (ST-ME-5)
Machine Learning Method (ME) Management (UT-ME)
Subscribe to updates (ST-CE-1)
Automated method deployment (ST-CE-2)
Monitor & terminate (ST-CE-3)
View & download results (ST-CE-4)
Train Model (ST-CE-5)
Track progress (ST-CE-6)
Customize processing pipeline (ST-CE-7)
Initialize training using snapshot (ST-CE-8)
Customize hyperparameters (ST-CE-9)
Computation Environment Managment (CE) (UT-CE)
Clinical Data Scientist
Medical Expert

FIGURE 4.2: Sub-Tasks: Persona centered overview

4.4.3 System-Functions and User Stories

The system-functions are needed to support the user tasks and subtasks listed in Table 4.4. This section describes the system-functions, which includes: the input data, output data, pre-conditions, post-conditions, exceptions, rules, the motivation of the user, and the role of the user. The order of the system-functions does not correlate with their priority.

Data Set Management

Data set management is one of the fundamental features of the platform. Management includes various tasks such as listing, uploading, updating, and removing data sets. The image and annotation format can vary depending on the task, the preference of the user, and on the source of the images. Therefore the platform should offer a generic solution, offering as much flexibility to the user as possible.

TABLE 4.9: System function: List data sets

Attribute	Description
ID	*REQ-1*, supports *ST-DS-1*
Pre-Condition	Platform is running and sample data sets are pre-loaded.
Input	Command to list the data sets.
Post-Condition	Nothing changed.
Output	Success or error message.

TABLE 4.10: System function: Import data set

Attribute	Description
ID	*REQ-2*, supports *ST-DS-2*
Pre-Condition	Data set *D* does not exist within the platform.
Input	The user uploads her data set using the web interface.
Post-Condition	Data set was successfully uploaded, basic metadata generated and updated.
Output	Success or error message.

TABLE 4.11: System function: Update data set

Attribute	Description
ID	*REQ-3*, supports *ST-DS-3*
Pre-Condition	Data set *D* is present and valid.
Input	Command to update data set *D*.
Post-Condition	Data set *D* is present in the platform.
Output	Success or error message.

TABLE 4.12: System function: Edit data set metadata

Attribute	Description
ID	*REQ-1*, supports *ST-DS-4*
Pre-Condition	Data set *D* is present and valid.
Input	Command to edit the data sets metadata *D*.
Post-Condition	The data sets metadata *D* is updated.
Output	Success or error message.

TABLE 4.13: System function: Remove data set

Attribute	Description
ID	*REQ-1*, supports *ST-DS-5*
Pre-Condition	Data set *D* is present and valid.
Input	Command to remove data set *D*.
Post-Condition	Data set *D* is removed.
Output	Success or error message.

Machine Learning Model Management

Management includes various tasks such as listing, importing, updating, and removing models. Machine learning models are developed with the aim to fulfill various tasks on matching the data sets. The structure, requirements, and functionality can vary considerably, making it a huge challenge to fit them into a single platform. Therefore the platform should attempt to offer a generic, standardized solution which provides as much freedom to the developer as possible.

TABLE 4.14: System function: List models

Attribute	Description
ID	*REQ-6*, supports *ST-MO-1*
Pre-Condition	Platform is running and ready to be used.
Input	Command to list the available models.
Post-Condition	Nothing changed.
Output	Success or error message.

TABLE 4.15: System function: Import model

Attribute	Description
ID	*REQ-7*, supports *ST-MO-2*
Pre-Condition	Model M does not exist within the platform.
Input	Reference to the model, which is accessible for the platform.
Post-Condition	Model was successfully imported and the metadata extracted and updated.
Output	Success or error message.

TABLE 4.16: System function: Update model

Attribute	Description
ID	*REQ-8*, supports *ST-MO-3*
Pre-Condition	Model M is present and valid.
Input	Command to update model M.
Post-Condition	Model M is updated.
Output	Success or error message.

TABLE 4.17: System function: Edit model metadata

Attribute	Description
ID	*REQ-9*, supports *ST-MO-4*
Pre-Condition	Model M is present and valid.
Input	Command to edit the model M metadata.
Post-Condition	Model M metadata is updated.
Output	Success or error message.

TABLE 4.18: System function: Remove model

Attribute	Description
ID	*REQ-10*, supports *ST-MO-5*
Pre-Condition	Model M is present and valid.
Input	Command to remove model M.
Post-Condition	Model M is removed.
Output	Success or error message.

TABLE 4.19: System function: List model snapshots

Attribute	Description
ID	*REQ-11*, supports *ST-MO-6*
Pre-Condition	Model M is present and valid.
Input	Command to list the snapshots of the model M.
Post-Condition	Nothing changed.
Output	Success or error message.

TABLE 4.20: System function: Remove snapshot

Attribute	Description
ID	*REQ-12*, supports *ST-MO-7*
Pre-Condition	Model M, Snapshot S is present and valid.
Input	Command to remove a snapshot S of model M.
Post-Condition	Snapshot S of model M is removed.
Output	Success or error message.

Machine Learning Method Management

Machine learning method management is one of the fundamental features of the platform. Management includes various tasks such as listing, creating, uploading patient data, and removing methods. Machine learning methods are pre-configured model snapshots which are able to fulfill various tasks on the uploaded patient data. The structure, requirements, and functionality can vary considerably, making it a huge challenge to fit them into a single platform. Therefore the platform should attempt to offer a generic, standardized solution, which provides as much freedom to the developer as possible.

TABLE 4.21: System function: List methods

Attribute	Description
ID	*REQ-13*, supports *ST-ME-1*
Pre-Condition	Platform is running and ready to be used.
Input	Command to list the available models.
Post-Condition	Nothing changed.
Output	Success or error message.

TABLE 4.22: System function: Create method

Attribute	Description
ID	*REQ-14*, supports *ST-ME-2*
Pre-Condition	Method *ME* does not exist within the platform.
Input	Command to create Method *ME*, including its configuration.
Post-Condition	Method was successfully created and is available in the platform.
Output	Success or error message.

TABLE 4.23: System function: Remove method

Attribute	Description
ID	*REQ-15*, supports *ST-ME-3*
Pre-Condition	Method *ME* is present and valid.
Input	Command to remove method *ME*.
Post-Condition	Method *ME* is removed.
Output	Success or error message.

TABLE 4.24: System function: Import patient data

Attribute	Description
ID	*REQ-16*, supports *ST-ME-4*
Pre-Condition	Analyze patient data is selected in the platform.
Input	Command to upload the locally available patient data archive *P*.
Post-Condition	The patient data *P* is updated, extracted and available for the method.
Output	Success or error message.

Computation Environment Management

The computation environment applies a machine learning model to a data set and guides the user through each step of the training setup. Due to the different kinds of models, the startup process needs to adapt to the selected model. After launching the training or inference, the platform monitors the process and collects important information and stores them in a standardized way. Therefore the platform should attempt to offer a generic, standardized solution which provides as much freedom to the model developer as possible.

TABLE 4.25: System function: Subscribe to process updates

Attribute	Description
ID	*REQ-17*, supports *ST-CE-1*
Pre-Condition	Process S is running within the platform.
Input	Subscription to process S, contact option Co.
Post-Condition	Process is monitored and platform sends updates to contact option Co.
Output	Success or error message.

TABLE 4.26: System function: Automated method application

Attribute	Description
ID	*REQ-18*, supports *ST-CE-2*
Pre-Condition	Patient Data PD and method ME is present.
Input	Command to apply method ME on patient data PD.
Post-Condition	A new result for method ME is present.
Output	Success or error message.

TABLE 4.27: System function: Monitor and terminate processes

Attribute	Description
ID	*REQ-19*, supports *ST-CE-3*
Pre-Condition	Processes $PR_1, \ldots, PR_N$ are running within the platform.
Input	Command to monitor these processes and command to delete processes $PR_1, \ldots, PR_m$.
Post-Condition	Information about the running processes are gathered and processes $PR_1, \ldots, PR_m$ were terminated.
Output	Success or error message.

TABLE 4.28: System function: Show and offer download of results

Attribute	Description
ID	*REQ-20*, supports *ST-CE-4*
Pre-Condition	Method Application MEA is finished and the results are available within the platform.
Input	Command to show the results of the method applications within the platform.
Post-Condition	Application results are stored and compressed to an archive, including a working download link for each result-archive.
Output	Success or error message.

TABLE 4.29: System function: Perform model training

Attribute	Description
ID	*REQ-21*, supports *ST-CE-5*
Pre-Condition	Data set D and model M is present.
Input	Command to train model M on data set D with configuration C.
Post-Condition	A new snapshot is present, containing the trained parameters and details of the training.
Output	Success or error message.

TABLE 4.30: System function: Track progress

Attribute	Description
ID	*REQ-22*, supports *ST-CE-6*
Pre-Condition	Training or Inference *TE* is created and still running.
Input	Command to track running Process *TE*.
Post-Condition	Nothing changed.
Output	Success or error message.

TABLE 4.31: System function: Customize processing pipeline

Attribute	Description
ID	*REQ-23*, supports *ST-CE-7*
Pre-Condition	Model M is present.
Input	Selected model M, selection to customize its pre- and post- processing pipeline *PI*.
Post-Condition	The customizable processing pipeline is applied during training.
Output	Success or error message.

TABLE 4.32: System function: Initialize training using model snapshot

Attribute	Description
ID	*REQ-24*, supports *ST-CE-8*
Pre-Condition	Model M is present and has a snapshot *PS*.
Input	Command to initialize model M with snapshot *PS* is given.
Post-Condition	Before training the model is initialized with snapshot *PS*.
Output	Success or error message.

TABLE 4.33: System function: Customize model hyperparameters

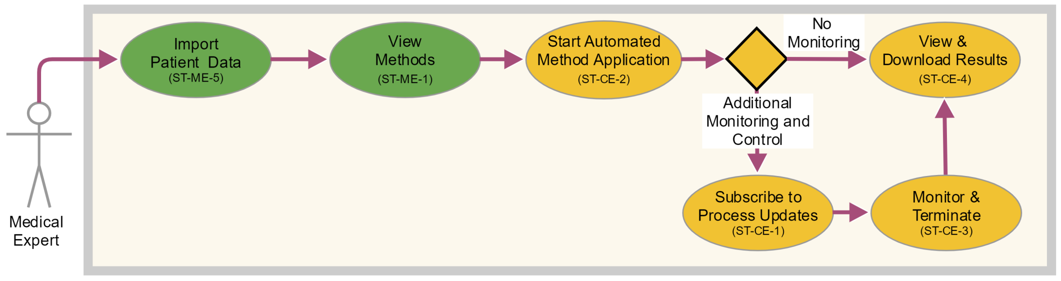
Attribute	Description
ID	*REQ-25*, supports *ST-CE-9*
Pre-Condition	Model M is present.
Input	Selected model M, selection to customize its configuration C.
Post-Condition	The customizable hyperparameters, default values and description is applied in the training of the model.
Output	Success or error message.

4.5 Single-User Workflow

The following section describes typical workflows for each persona, i.e., in which sequence the user would perform the sub-tasks mentioned earlier. The workflows introduced in the section represent the most common use cases for each persona.

4.5.1 Medical Expert

The medical expert's main motivation is to apply state-of-the-art machine learning methods to her patient images without needing a machine learning background. Usually, the methods annotate specific areas in the images, such as cancer locations to support the medical expert with the image analysis. A generalized workflow is shown in Figure 4.3



Medical Expert
Import Patient Data (ST-ME-5)
View Methods (ST-ME-1)
Start Automated Method Application (ST-CE-2)
No Monitoring
View & Download Results (ST-CE-4)
Additional Monitoring and Control
Subscribe to Process Updates (ST-CE-1)
Monitor & Terminate (ST-CE-3)

FIGURE 4.3: Activity diagram for the medical expert

- Persona: **Medical Expert**
- Default use-case and platform behavior (as shown in Figure 4.3)
  1. Upload patient data *(ST-ME-5)*
     Patient data is uploaded, imported, and stored in a temporal directory within the platform.
  2. View, search, filter and select methods *(ST-ME-1)*
     All available methods are listed, a search function and filter are available to help the user to find the needed methods. Details about the methods are displayed on request, and system attributes are hidden from the user.
  3. Deploy automated inference using selected methods *(ST-CE-2)*
     The pre- and post-processing pipeline and all hyperparameters are already pre-defined in the method. After pre-processing, the trained model is inferred on the medical data. The results of the inference are processed by the post-processing pipeline and stored in a persistent storage on the platform.
  4. Access and download results *(ST-CE-4)*
     The user receives a notification about the available results, including a link to download them as a zip-archive. The archive contains data and information related to the inference, including the logs of the pipelines and source images.

- Quick-deployment use-case
  1. Selection of a method *(ST-ME-1)*
  2. Use corresponding quick-deployment button
  3. Import of patient data *(ST-ME-5)*
  4. Deploy automated inference *(ST-CE-2)*

4.5.2 Clinical Data Scientist

The main motivation for the clinical data scientist is to improve and adapt state-of-the-art machine learning models to specific tasks and share them with other users. To optimize the machine learning method for a specific task, she uses a range of data sets, modifies the model architecture, and tunes its hyperparameters. When the clinical data scientist is satisfied with the model's performance, she generates a model snapshot and, additionally, an automated method. The method can now be conveniently shared with clinical data scientists, who hence have easy access to the latest developments of the machine learning model. A generalized workflow is shown in Figure 4.4.

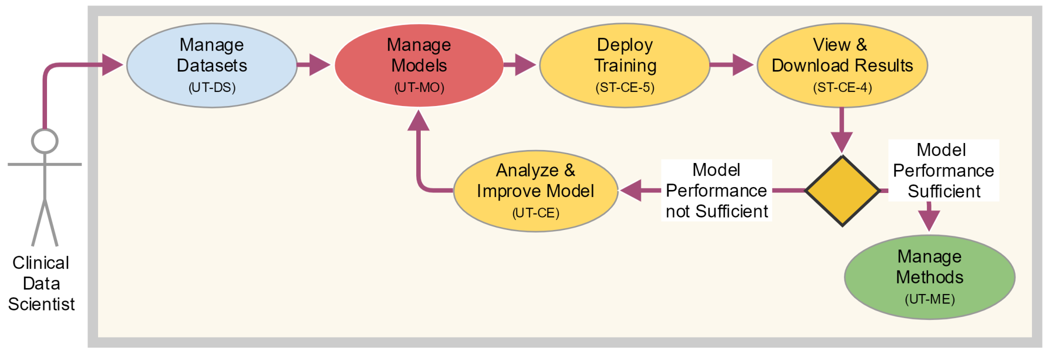

Manage Datasets (UT-DS)
Manage Models (UT-MO)
Deploy Training (ST-CE-5)
View & Download Results (ST-CE-4)
Model Performance Sufficient
Model Performance not Sufficient
Analyze & Improve Model (UT-CE)
Manage Methods (UT-ME)
Clinical Data Scientist

FIGURE 4.4: Activity diagram for the clinical data scientist

- Persona: **Clinical Data Scientist**
- Default use-case and platform behavior (as shown in Figure 4.4)
  1. Upload data set *(ST-DS-2)*
     The data set is uploaded and imported from an archive of the images and labels. The data set is stored in a persistent storage location on the platform.
  2. Import model *(ST-MO-2)*
     Models are imported from a for the platform accessible repository. Needed attributes are collected and stored. The model is stored in a persistent storage location on the platform.
  3. Train, the model on a selected data set *(ST-CE-5),*
     The platform guides the user through the training process. This includes

steps such as selecting a previous model snapshot to start the training with, customization of available hyperparameters for the selected model and monitoring.

4. Update model version *(ST-MO-3)*
   The user changes the source code of the model and adds a new tag within the source code repository. During update the platform detects the new version and updates the model version accordingly.
5. Train updated model *(ST-CE-5)*
   The platform adds a new model snapshot to the updated model. Snapshots of the old version can not be used anymore as a starting point for a new training.
6. Improve trained updated model *(ST-CE-8, ST-CE-9, ST-CE-5)*
   A new training is launched, using the snapshot of the previous training as a starting point of the training.
7. Access and download training results *(ST-CE-4)*
   The user accesses the training results by downloading a zip-archive with all data related to the training (including logs and statistics).
8. Create method from model and a selected snapshot *(ST-ME-2)*
   The user exports the selected snapshot as a method, by adding a simple name, description, and pre- and post-processing pipeline.

- Remove unused or old methods *(ST-ME-3)*,
  The user removes methods that are not used anymore from the platform.

4.6 Multi-User Collaboration Workflow

One of the main goals of this platform is to connect medical experts to the clinical data scientists to provide them with state-of-the-art methods for medical data analysis.

A generic sample workflow is assumed to involve three parties, from which one focusses on the medical tasks, and two are focussing on machine learning related research in the medical context.

The collaboration requires shared access to data and machine learning research. The following list shows an overview of the most common concerns we gathered during interviews with potential users:

- Legal Restrictions
  - Sensitive patient data can only be accessed within a strictly protected environment.
  - Sensitive data sets are used to train machine learning models without exposing the data to the users.
  - The platform should be aware of sensitive patient data users may upload during the setup of the analysis workflow. After a method was trained using sensitive data, it may also be sensitive itself.
- Political Restrictions

  - Accumulating high-quality data for research is costly and time-consuming. Therefore some researchers may not want to share the data freely with others but allowing other users of the platform to use it in a controlled way under their supervision.

- Technical Restrictions
  - Required data is too large to copy it over the network in a reasonable time.
  - Required data is too large for storing it on the target platform.

Therefore the platform provides different handling of data, depending on the purpose. To develop new state-of-the-art models, anonymized or insensitive data sets are stored and shared within the platform. For applying machine learning methods to sensitive patient data, the data is collected and only used for one application. To support this concern, they need multiple computation environments, which are used to train models or deploy newly developed methods. Therefore the computation environment should be standardized and not depend on the other parts of the platform. The following scenario describes the interaction between the users:

1. Medical experts are working with large sets of patient data, and they need to analyze it to gain further insights on their data. Currently, they analyze the data manually but wonder whether there are state-of-the-art methods available to reduce their workload.
2. The medical experts contact a clinical data scientist CDS_1 to evaluate the possibility of leveraging machine learning concepts to analyze the patient data.
3. CDS_1 explores currently available machine learning models using state-of-the-art publications, her own research, and research from other clinical data scientists she works with.
4. A clinical data scientist CDS_2 mentioned in a previous meeting that she has developed a machine learning model solving a similar problem and is interested in customizing the model to meet the needs of the medical experts.
5. CDS_2 uploads her current model into the medical machine learning platform.
6. CDS_1 and CDS_2 are working together, understanding how the model works and which steps are needed to customize it for the medical experts.
7. They customize and improve the machine learning method iteratively using the platform, based on available data sets and sample data sets provided by the medical experts for this specific issue.
8. After some time they finalized the model and use it to create a new method. The method automates the application of the model and makes it straightforward for the medical experts to apply it to their patient data.
9. They notify the medical experts that the method is available in the platform.
10. The medical experts apply the method to their patient data.

The platform focuses on the machine learning aspects. Figure 4.5 shows all and actors and possible interaction points. The medical experts wish to leverage modern machine learning approaches to reduce their manual workload. Therefore, they ask a clinical data scientist CDS_1, if it is possible to develop a new method for their use case. Additionally, they provide insensitive sample data, which is accessible by the clinical data scientists. Data sets are stored in the platform and managed by the clinical data scientists. Due to the complexity, the figure focuses on the model and method development. The grayed tasks are not part of the platform but may be helpful to understand the workflow.

Researcher CDS_1 decides whether to collaborate with other researchers. They develop a new model iteratively and create an automated method afterward. The method is mainly accessed and used by medical experts. In the ideal case, they can load their sensitive data directly into the computing environment and apply the new method there. This way, the other parties will never have access to the real data but can get valuable information about the performance later. In the future, this behavior could also be used to train machine learning models on real patient data without providing direct access to the researchers.

Another important aspect of the platform is to share and provide access to valuable data sets. The creation of high-quality data sets is usually very costly and may contain sensitive information. Therefore it is not always possible to make it publicly available. This platform should store these data sets and allows other users to use them for training their model. The focus is on the general functionality of managing data sets and using them to train machine learning models within the platform. The big advantage in comparison to publishing the data set is in the control of who uses it and how. The access could be revoked any time, and changes can also be easily applied.

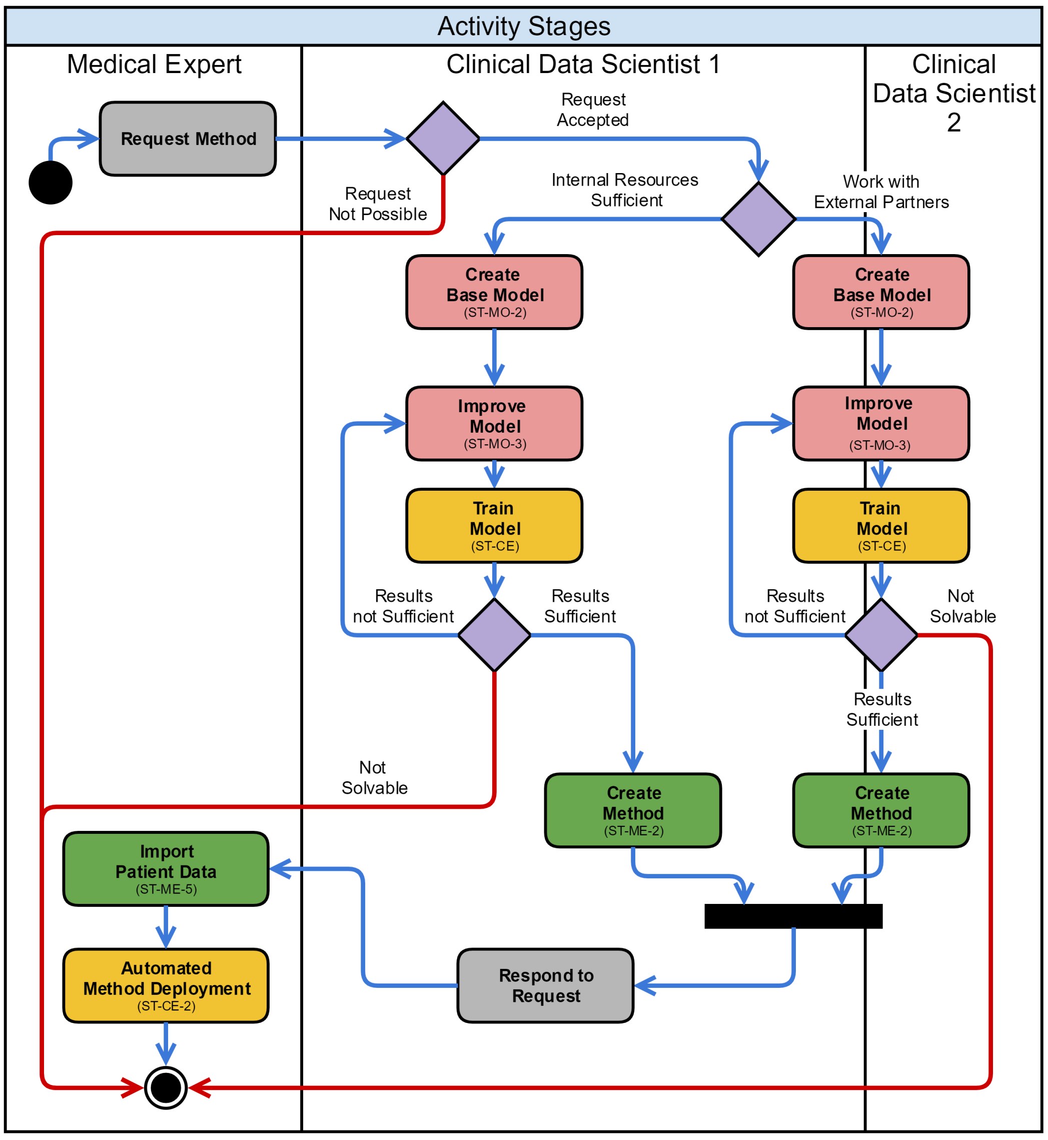


FIGURE 4.5: Multi-user collaboration workflow design

4.7 Non-Functional Requirements

Non-functional requirements (NFR) describe the non-behavioral attributes of a system, including completeness, correctness, efficiency, interoperability, usability, reliability, maintainability, portability, test-ability, and understandability [23]. The categories used to describe the NFR are derived from [27]. These are described for this platform in the following Tables: sustainability Table 4.35, and Table 4.36, performance efficiency in Table 4.37, compatibility and interoperability in Table 4.38, usability in Table 4.39, reliability in Table 4.40, maintainability in Table 4.41, and Table 4.42. A full overview of all functional and non-functional requirements is shown in Table 4.43.

TABLE 4.34: NFR: Functional compatibility

Attribute	Description
Category	Functional Suitability
Description	This platform should provide very flexible management for data sets and models to minimize the burden of using the platform. The compatibility of different kinds of data sets and models is a necessary requirement for medical researches.
Criteria	Data sets containing various types of data, such as images, audio, text, and videos are supported. Machine learning models are not restricted to specific frameworks.
Time	Design and Development-Stage

TABLE 4.35: NFR: Functional completeness

Attribute	Description
Category	Functional Suitability
Description	The developed platform is fulfilling the desired requirements.
Criteria	The defined system-functions are implemented.
Time	Design and Development-Stage

TABLE 4.36: NFR: Functional correctness

Attribute	Description
Category	Functional Suitability
Description	The developed platform was tested to assure the quality and correctness of each component. The implementation adapts to design decisions such as the General Responsibility Assignment Software Patterns (GRASP) [86]. These patterns include high cohesion, low coupling, polymorphism, protected variations, indirection, and pure fabrication.
Criteria	Test strategies are described.
Time	Design and Test-Stage

TABLE 4.37: NFR: Time-behavior

Attribute	Description
Category	Performance Efficiency
Description	The user interface should respond quickly, with the exception of loading a large data set or using a complex machine learning method.
Criteria	On average the response time should be less then 1 second.
Time	Design- and Development-Stage

TABLE 4.38: NFR: Interoperability

Attribute	Description
Category	Compatibility
Description	The platform is fully containerized and deployed within a fully virtualized container environment.
Time	Design- and Development-Stage
Criteria	Platform is containerized.

TABLE 4.39: NFR: User-interface

Attribute	Description
Category	Usability
Description	User feedback is displayed in case of issues after the user inputs data or uses an advanced functionality of the platform.
Time	Development- and Test-Stage
Criteria	A notification is displayed after invalid user input.

TABLE 4.40: NFR: Fault tolerance

Attribute	Description
Category	Reliability
Description	The platform tolerates invalid user input. If the user provides invalid information, the platform should display a notification. After that the user should be able to correct her input.
Time	Development- and Test-Stage
Criteria	The platform does not crash after invalid user input is provided.

TABLE 4.41: NFR: Modularity

Attribute	Description
Category	Maintainability
Description	The developed platform uses design patterns, separation of concerns, and matching design structures.
Time	Design- and Development-Stage
Criteria	Each system component runs separately from the others.

TABLE 4.42: NFR: Reusability

Attribute	Description
Category	Maintainability
Description	The developed platform allows adding more components later.
Time	Design- and Development-Stage
Criteria	The source code is structured in a way that enables others to modify each component easily.

4.8 User-Interface Structure Design

The user interface structure diagram describes the interaction between the user and the platform. Workspaces describe which content is shown and the system-functions used to present it. Arrows between the workspaces indicate how the user can navigate between each of them. Sophisticated workspaces are divided into smaller sub-workspaces. Special sub-workspaces are further separated and shown in a different diagram.

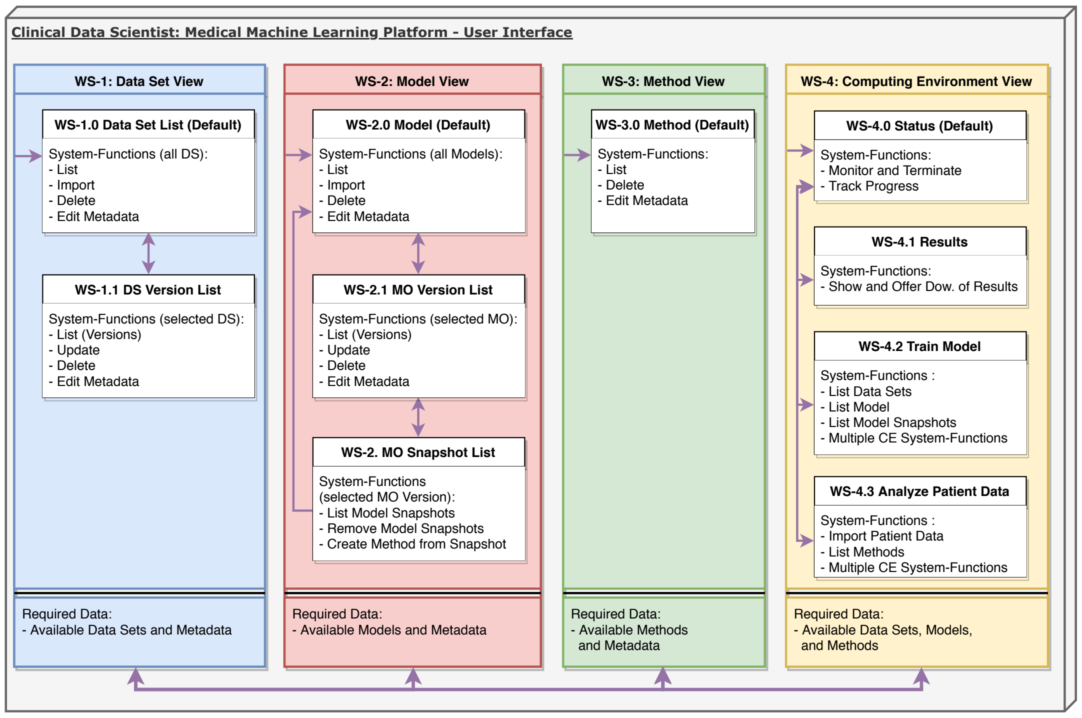


FIGURE 4.6: User interface structure diagram for the clinical data scientist

The platform offers multiple workspaces based on the persona who is using it. If the clinical data scientist is using the interface, workspaces one to 4 are used. They are shown in Figure 4.6. *WS-1* shows the workspace related to the data sets. If the user enters the workspace, *WS-1.0* is shown. It offers a list of all data sets, uploading, deleting, and editing of the metadata. To further inspect a data sets, *WS-1.1* offers a view about the available versions and the option to update a new version. *WS-2* focusses on the available models. Per default, *WS-2.0* is shown, which provides a list and functions to manage the models. *WS-2.1* shows the available versions of a selected model and offers additionally the possibility to view and remove model snapshots and create methods by navigating to workspace *WS-2.2*. *WS-3* and *WS-3.0* offers to list, delete, and edit the metadata of available methods. *WS-4* guides to all the needed steps for training a model on a data set and applying a method to uploaded patient data. It defaults to *WS-4.1*, which shows the currently running processes, options to terminate them and track the progress. To check and download the results of completed method applications, the user can navigate to *WS-4.1*.

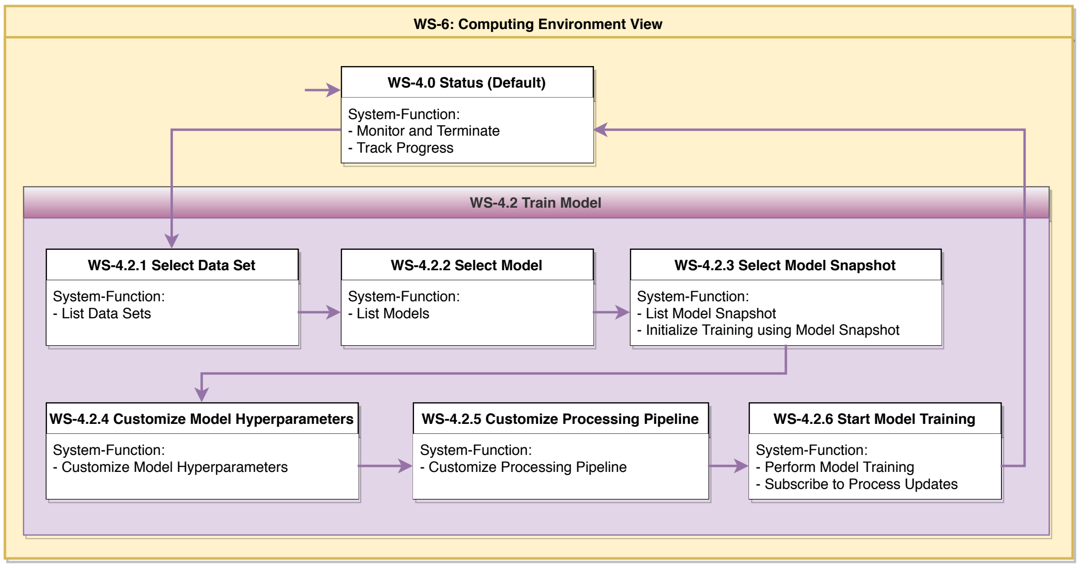


FIGURE 4.7: User interface structure diagram for model training

Training a model requires a large amount of customization. The interface guides the user through every step in *WS-4.2*. Due to the high complexity the steps are shown in Figure 4.7. The first step is to select a data set in *WS-4.2.1*. In the second step *WS-4.2.2* the user selects the model to train. Sometimes it is required to improve a previous training result further. Therefore *WS-4.2.3* lists all available snapshots for the selected model version. In *WS-4.2.4*, the hyperparameters of the model are displayed with the options to customize them. Depending on the model, it may be required to adjust the processing pipeline (pre and post-processing). *WS-4.2.5* shows the default pipeline and allows to modify it. In the last step *WS-4.2.6*, a summary of the configuration is shown, and the user can start the training of the model. After the training is started, the progress is shown in *WS-4.0*. To test newly created methods, *WS-4.3* offers to apply methods to patient data, as the medical expert would do. The workspace is the same as described in *WS-6.3* and shown in Figure 4.9.

In contrast to the clinical data scientist, the medical expert focuses on applying methods to patient data. Therefore the interface differs a lot. The platform should offer *WS-5* and *WS-5.1* to display the available methods to the user. To apply one of them to the patient data, the computing environment *WS-6* is required. *WS-6.1* welcomes the user with an overview of the currently running method applications. The user can navigate to *WS-6.1* to view and download old results or start an analysis of her patient data. Due to the complexity of the analysis, this process is shown in Figure 4.9

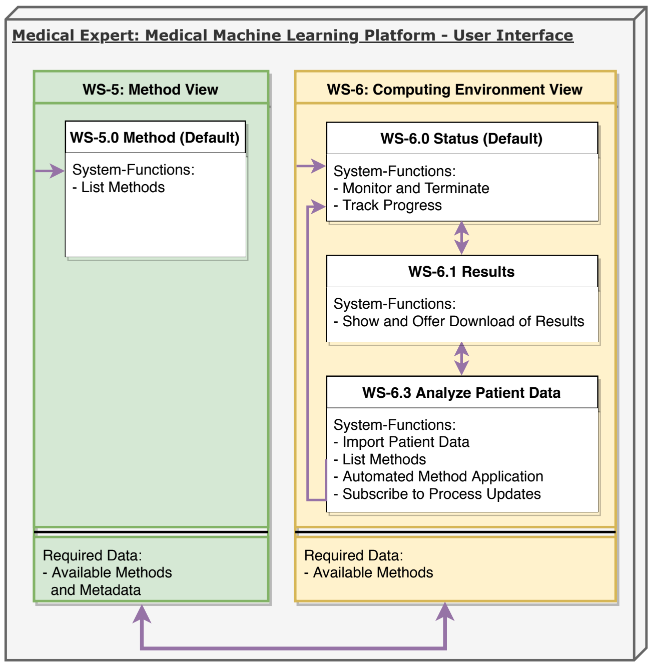


FIGURE 4.8: User interface structure diagram for the medical expert

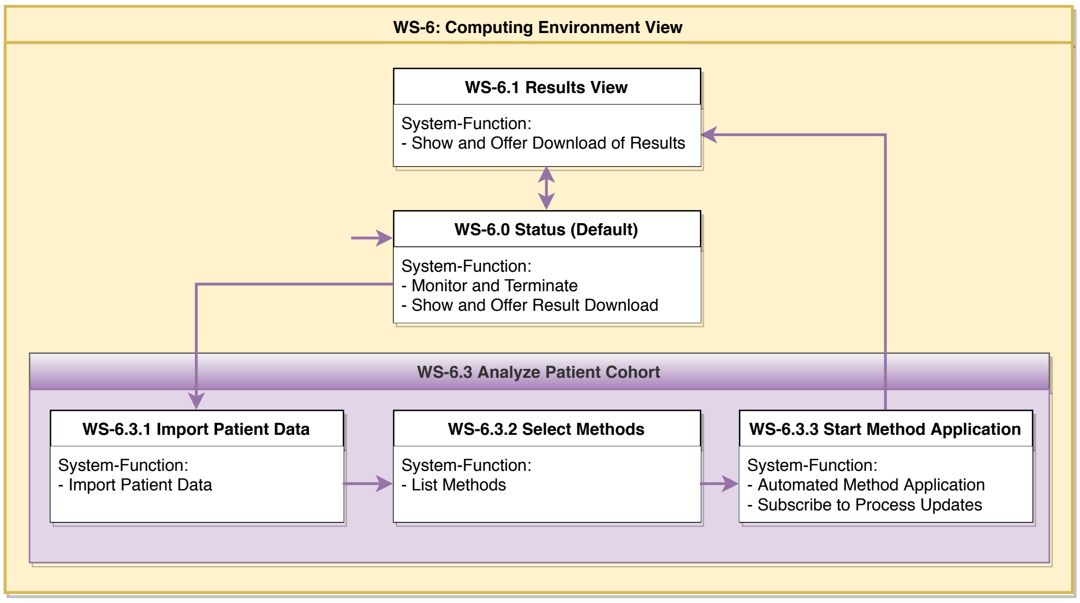

WS-6: Computing Environment View
WS-6.1 Results View
System-Function:
- Show and Offer Download of Results
WS-6.0 Status (Default)
System-Function:
- Monitor and Terminate
- Show and Offer Result Download
WS-6.3 Analyze Patient Cohort
WS-6.3.1 Import Patient Data
System-Function:
- Import Patient Data
WS-6.3.2 Select Methods
System-Function:
- List Methods
WS-6.3.3 Start Method Application
System-Function:
- Automated Method Application
- Subscribe to Process Updates

FIGURE 4.9: User interface structure diagram for the method application

The first step of analyzing a patient data is to upload it into the platform. Uploading the patient data is possible in Figure 4.9. In the next step, the medical expert has to view all available methods and select the method required to gain the needed insights on the patient images, which is provided by *WS-6.3.2*. Lastly, *WS-6.3.3* provides the option to review the configuration before launching it. Also, some advanced features such as a subscription to updates can be selected to notify the users if certain milestones are reached. After the application finished, the user can access the results page to view and download the produced results.

4.9 Takeaway

The personae who are using the platform were defined in Section 4.2. A primary persona is the *Medical Expert*, as described in Table 4.1. The medical expert works in the medical field and wants to leverage machine learning approaches to support her during patient image analysis. Another primary persona is the *Clinical Data Scientist*. The clinical data scientist is defined in Table 4.2.

She has a much broader scientific background, works with machine learning regularly, and wants to use the platform to develop and share her machine learning models. All use cases are described in Table 4.4. These use cases were used to derive functional requirements, which are described in Section 4.4, the user workflows in Section 4.5, multi-user workflows in Section 4.6, and non-functional requirements are described in Section 4.7. The user interaction with the platform is described in Section 4.8. An overview of all requirements is shown in Table 4.43.

TABLE 4.43: Requirements analysis: Requirements overview

ID	Type	Name
Data Set Management:		
REQ-1	FR	List Data Sets (Table 4.9)
REQ-2	FR	Upload Data Set (Table 4.10)
REQ-3	FR	Update Data Set Version (Table 4.11)
REQ-4	FR	Edit Data Set Metadata (Table 4.12)
REQ-5	FR	Remove Data Set (Table 4.13)
Model Management:		
REQ-6	FR	List Models (Table 4.14)
REQ-7	FR	Import Model (Table 4.15)
REQ-8	FR	Update Model (Table 4.16)
REQ-9	FR	Edit Model Metadata (Table 4.17)
REQ-10	FR	Remove Model (Table 4.18)
REQ-11	FR	List Model Snapshots (Table 4.19)
REQ-12	FR	Remove Snapshot (Table 4.20)
Method Management:		
REQ-13	FR	List Methods (Table 4.21)
REQ-14	FR	Create Method (Table 4.22)
REQ-15	FR	Remove Method (Table 4.23)
REQ-16	FR	Import Patient Data (Table 4.24)

table continued on next page

Table 4.43 — continued from previous page

ID	Type	Name
Computation Environment:		
REQ-17	FR	Subscribe to Process Updates (Table 4.25)
REQ-18	FR	Automated Method Application (Table 4.26)
REQ-19	FR	Monitor and Terminate Processes (Table 4.27)
REQ-20	FR	Show and Offer Download of Results (Table 4.28)
REQ-21	FR	Perform Model Training (Table 4.29)
REQ-22	FR	Track Progress (Table 4.30)
REQ-23	FR	Customize Processing Pipeline (Table 4.31)
REQ-24	FR	Initialize Training using Model Snapshot (Table 4.32)
REQ-25	FR	Customize Model Hyperparameters (Table 4.33)
Non-Functional Requirements:		
REQ-26	NFR	Functional Suitability: Functional Compatibility (Table 4.34)
REQ-27	NFR	Functional Suitability: Functional Completeness (Table 4.35)
REQ-28	NFR	Functional Suitability: Functional Correctness (Table 4.36)
REQ-29	NFR	Performance Efficiency: Time-behavior (Table 4.37)
REQ-30	NFR	Compatibility: Interoperability (Table 4.38)
REQ-31	NFR	Usability: User-Interface (Table 4.39)
REQ-32	NFR	Reliability: Fault tolerance (Table 4.40)
REQ-33	NFR	Maintainability: Modularity (Table 4.41)
REQ-34	NFR	Maintainability: Reusability (Table 4.42)

Chapter 5

Design and Implementation

This chapter describes the fundamental design decisions and the implementation of the prototype.

5.1 Overview

The overall goal is to integrate and standardize machine learning research approaches into one platform. With this prototype, we aim to provide a proof of concept for the fundamental idea behind the platform: building a scalable collaboration hub for clinical data scientists and medical experts, in which they can train, share, and apply machine learning models. Thus, the first effort was focussed at the essential functionality of the platform: model training and model application. The concept and implementation work can be subdivided into four parts: frontend, backend, computing environment, and infrastructure. Special attention was directed at the underlying infrastructure to support on-demand scaling and high computation power. Each component of the platform should follow a cloud-native design approach, as described in Section 2.3 to exploit the advantages of cloud computing, such as notably better scaling capabilities. The design of the platform prototype includes the full infrastructure stack and therefore can adapt to the specialties of a cloud compute cluster. A local workstation, supporting the development and the execution of virtualized application containers, is used as a simplified representation for the prototype. In further work, the platform can be, as it is built on top of scalable infrastructure, integrated into the proposed fully capable cloud computing cluster environment.

General Architecture

The platform consists of thee major components, which run on top of the infrastructure: the frontend, the backend, and the computing environnement (Figure 5.1). The user has access to the frontend to manage data sets, models, and methods. When the user uses the frontend, requests are created and send to the backend. The backend processes the user requests and executes corresponding function calls. Computing jobs are run inside a fully virtualized computing environments, which can theoretically be deployed in various locations (e.g., local machine, a compute cluster, or a cloud provider).

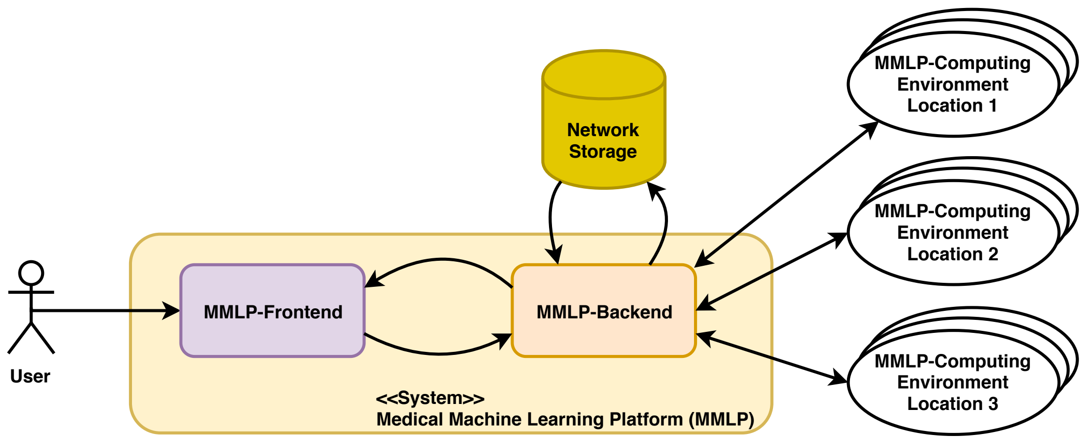

Network Storage
MMLP-Computing Environment Location 1
MMLP-Computing Environment Location 2
MMLP-Computing Environment Location 3
MMLP-Frontend
MMLP-Backend
User
<<System>>
Medical Machine Learning Platform (MMLP)

FIGURE 5.1: Main components of the platform

5.2 Frontend

Both medical experts and data scientist have to work on the project. Therefore, the user interface should provide access for multiple users from separate locations. Due to the various types of workplaces in the medical field, it also cannot be assumed that the local computer is powerful enough to allow support machine learning-related tasks.

Therefore, we have chosen to employ a web-based frontend to provide access to the platform. The web-based frontend will utilize a backend to process the user requests and multiple computing environments, which performs the training of a model and application of a method. Opposed to an application-style frontend, a web-based frontend imposes minimal soft- and hardware requirements. Furthermore, it does not need local installations, and users can easily access the platform using an internet browser.

We developed the frontend following a user-centred design approach, focussing on the users' needs during every step of the development [33]. The design components include how data and functions are represented, how the layout of the screen is structured, and how the dialogs are described. In this work, two personae are considered: the medical expert (described in Table 4.1) and the clinical data scientist (described in Table 4.2). As shown in Section 4.5, the personas have different tasks that the platform either enables or supports. Additionally, the tasks may require very different data, information, and functionalities. Therefore the interface should distinguish between each persona, having an option to select which persona wants to use the platform. To further support the personas, the interface should be simplified and automated as much as possible. For clinical data scientists, a simple but powerful frontend is required, focusing on flexibility and an extensive feature-set, without sacrificing ease-of-use and intuitiveness. Further, clinical data scientists should be able to upload annotated data to train a segmentation model with minimal restrictions on the format, structure, or other aspects.

The focus for the medical expert was to provide an accessible, intuitive, and modern interface to additionally motivate the usage of a new system. To use an exported method with patient data, the interface should provide clear guidance, following a three-step approach: upload data, select method, and start the analysis process. Further configuration should not be necessary, because all relevant information was already collected during training. Additionally, the interface should be intuitive to use and aesthetically pleasing, to take advantage of the platform without requiring training classes, especially for the users without a computer science background.

A training pipeline could take multiple days to complete, during which the platform has to stay responsive to the users. Therefore, time-intensive functions should run asynchronously, enabling the user to use other functions while parts of the system are busy. Additionally, the platform should be able to send notifications to inform the users about important changes, such as finished training or application pipelines.

Frontend Implementation The user interface is build upon Node.js and JavaScript [84]. Node.js is a framework to develop high-performance, concurrent web applications and used for the frontend of the platform. On top of Node.js, the JavaScript Framework Vue.js [57] is used to implement the required functionalities. In general, Vue.js offers the following advantages: it supports a stateless and instanceless functional developing model, it has already a variety of standard components available, it has a variety of user libraries available, and it offers the capability to implement a android app in the future [90]. During the implementation process of the platform, we utilized the user-made material component framework Vuetify.js [91].

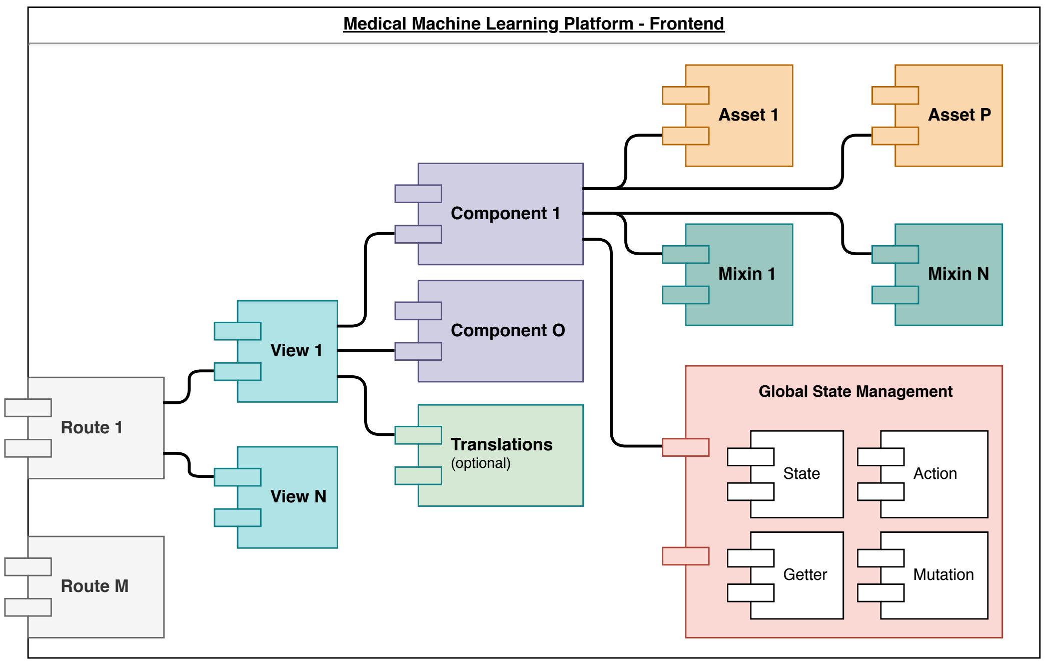


FIGURE 5.2: Frontend: Generic component diagram

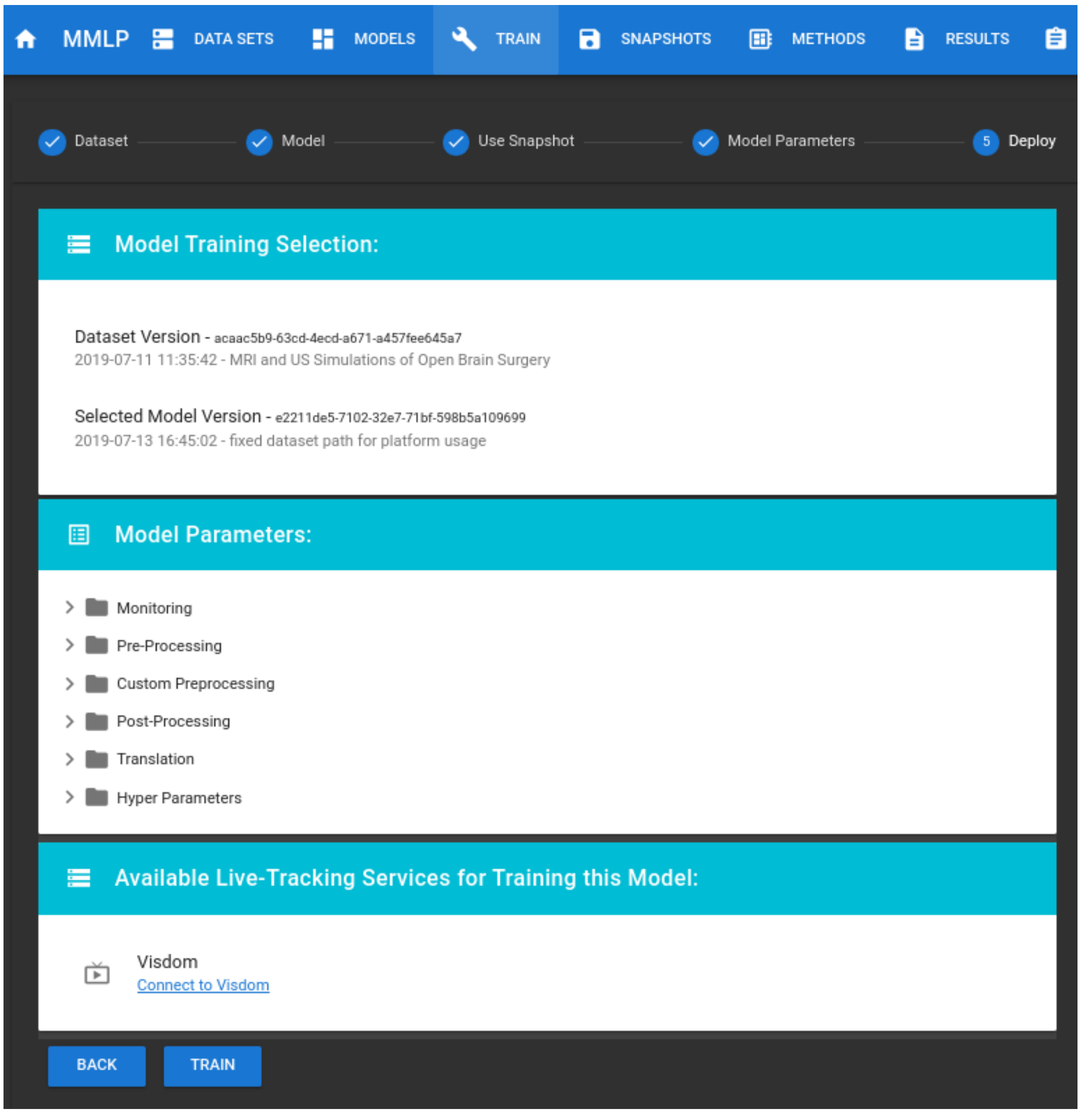
The frontend requires multiple components to process and display the requested pages from the user, which are shown in Figure 5.2. The content is divided into routes, which are defined by the Uniform Resource Locator (URL). Based on the

URL, the main application of the frontend forwards the request to the configured view or displays an error if no view is defined. An extemporary URL could be *'https://MMLP.de/datasets'*. The router would call and show the datasets overview view. Each view could be described as a dynamic page of the website, which adapts its content according to the provided data. The content is displayed using components which define how the data is presented and which functionalities are available to the user. A simple component could be the data set overview table, including the functions such as upload, add version, delete, and edit the data set. Each component can require additional functions to process and display the data. To use a function multiple with different components, they are shared using a *mixin* component. Similar to mixins, asset components contain objects, for example, static pictures, which can be shared by multiple frontend components. Notably, instead of implementing a synchronous parent-child-tree communication, the platform utilizes asynchronous communication with a global state management component. Asynchronous communication reduces code complexity and increases the responsiveness of the user interface.

The global state management component consists of four major parts: a getter, state, action, and mutation component. The state component keeps track of the current value of the data. The getter component is used to read data from the store. Specifically, getter components are functions that can access the data and apply arbitrary functions to it before the data is returned, for example, filters or extraction of specific parts. Two components are available to modify the content of the state: action and mutation. The critical difference between actions and mutations is that an action runs asynchronously, and is thus not blocking the frontend while it is running. Therefore the communication with the backend always uses action components, allowing the user to still use the frontend while data is sent or received from the backend. Mutation components are only used in the frontend for internal modifications of its data, which are usually very fast and thus does not notably block the user interface.

A sample screenshot of the frontend is shown in 5.3. The menu-toolbar shows available options, customized to the persona using the platform. From left to right, the menu includes the following features: data set management, model management, start training workflow, snapshot management, method management, result management, and a switch to change the persona.

The screenshot displays the summary page in the training workflow of the clinical data scientist. The summary page lists the complete configuration of the training process. In this case, the data set, model version, set model parameters, and available live-tracking services for training the models are presented. The small check-marks next to the menu-toolbar indicate the current state in the workflow. In this example, she already successfully configured the data set, model, snapshot, and model parameters. In case of a faulty configuration, an error sign is displayed, and the user is not able to proceed until the error is fixed.



Model Training Selection:

Dataset Version - acaac5b9-63cd-4ecd-a671-a457fee645a7
2019-07-11 11:35:42 - MRI and US Simulations of Open Brain Surgery

Selected Model Version - e2211de5-7102-32e7-71bf-598b5a109699
2019-07-13 16:45:02 - fixed dataset path for platform usage

Model Parameters:

- Monitoring
- Pre-Processing
- Custom Preprocessing
- Post-Processing
- Translation
- Hyper Parameters

Available Live-Tracking Services for Training this Model:

Visdom
Connect to Visdom

FIGURE 5.3: Frontend sample screenshot

5.3 Backend

The backend is used to process user requests from the frontend with the underlying programs of the platform and exposes the API to other programs. Exposing the API allows other programs to access the platform functionality directly. This offers further integrateability to other tools such as image viewer and machine learning related tools.

The backend implements the functional requirements mentioned in Chapter 4, especially *REQ-1* to *REQ-16*. An especially important task is the execution of training and application of machine learning models in the computing environment, where the data related to user-specified input has to be routed to the corresponding container. We have chosen Python as the employed programming language, as it is frequently used in the scientific field and therefore could encourage other collaborators to join the platform development in the future [71]. The main contracts in Python are object-oriented and also supports some of the functional programming approaches. The advantage of functional programming is an effective and elegant programming style which facilitates the rapid development of new functionalities. This is also essential for quality assurance, which is further described in Chapter 6.

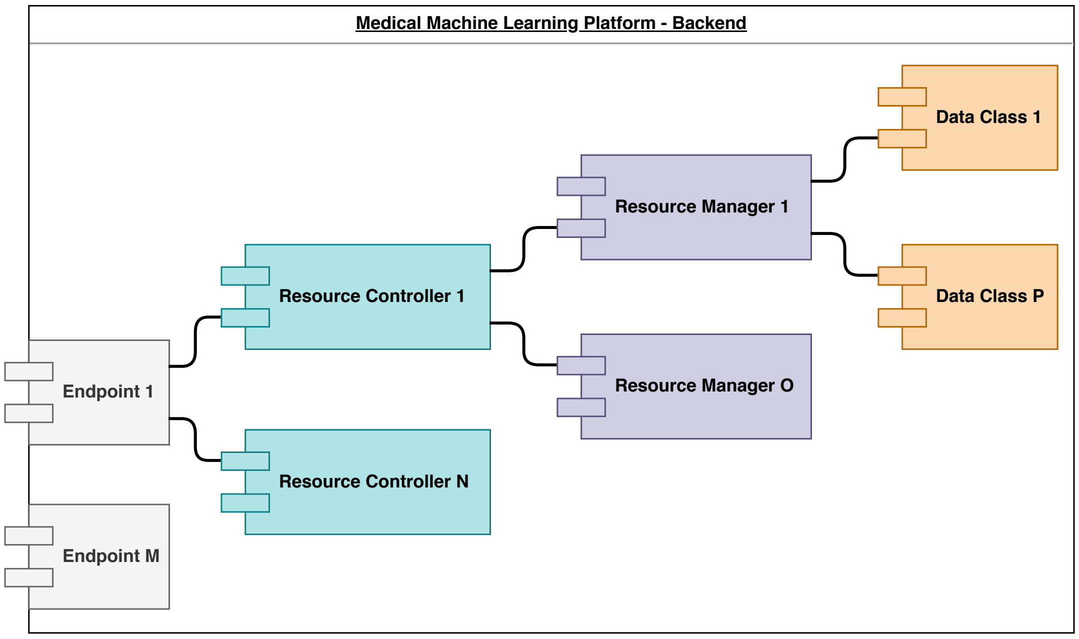


FIGURE 5.4: Backend: Generic component diagram

Backend Implementation The structure of the backend service is shown in Figure 5.4. Incoming requests have to be forwarded to the endpoint for processing, which is defined in the right part of the diagram. A request has an action, such as GET, POST, or DELETE, and a resource identifier, which is part of the Uniform Resource Locator (URL). Based on the resource identifier, the matching endpoint is selected and the matching resource controller called. The resource controller validates the requests, performs the required steps before it calls the resource manager(s) to provide the requested functionality. The resource manager has access to the data classes and the functions required to work with and manipulate the stored data.

For example, if the user sends a GET request to *'https://MMLP.de/datasets'*, the resource id *datasets* is extracted, and passed to the resource controller. The resource controller validates the input, and calls the resource *data-set-manager* to collect a list of all available data sets. In the last step, the data set resource manager extracts the required information from the stored data-set-classes and returns it to the controller, which creates the required http-response and sends it back to the user.

The API Falcon [35] is used to realize the backend. Compared to other frameworks, Falcon is rather minimalistic, robust, performant, and only requires a small amount of dependencies [35].

5.4 Computation Environment

The computation environment handles all tasks related to defining, running, monitoring, and controlling training of models, and application of methods. It is closely connected to the backend and implements *REQ-17* to *REQ-25*. There are various options available to implement such an environment. For specific frameworks such as PyTorch [48], a python based virtual environment could be used. However, this only applies to certain frameworks and is not a programming language-independent solution. Therefore, to support more generic computation environments system level virtualization strategies such as application containers are recommended. Application containers support nearly all kinds of programs, allowing even non-machine learning models to be used within the platform.

Additionally, the required computing power to develop and apply new machine learning methods has to be provided and will potentially grow over time. Further, contrary to the clinical data scientists, the compute capability of the local medical experts' workstation may not be sufficient for modern machine learning methods, especially when running multiple training pipelines in parallel. Therefore, the platform should provide its own infrastructure stack such as a GPU compute cluster on-premise or in the cloud. Therefore, the design should consider scaling capabilities of the infrastructure and be able to handle large amounts of data, high demand of CPU and GPU-computing power, and the parallel execution of independent model training and method analysis pipelines. This could be tackled by using high-end local workstations or local compute clusters. Other disciplines, for example in the field of bio-informatics, already use web-based cloud solutions run large, on-demand compute clusters [53]. Cloud solutions can be on-premise, managed by a local service provider, publicly by common providers such as Amazon Web Services or Google Compute Cloud, or a hybrid solution using both of them. This trend is also emerging in the medical field [64, 29]. Therefore the foundation of this platform is designed to support local, public and hybrid cloud solutions. Building on this cloud concept, each component of the platform is running in its own virtualized environment, independently from the hardware or the system its running on. This is further described in Section 2.3.

Model Training Process

The model training process is handled in large parts by the computation environment. In Figure 5.5 the communication of the system components and required steps to train a model are shown.

The clinical data scientist uses the frontend to manage data sets (REQ-1, REQ-2, REQ-3, REQ-4, and REQ-5) and machine learning models (REQ-6, REQ-7, REQ-8, REQ-9, REQ-10, REQ-11, and REQ-12).

First, she selects a data set version and the model version she wants to use. Then, to start the training, she configures training hyperparameters (REQ-25), customizes the processing pipeline (REQ-23), optionally initializes the training with a model snapshot (REQ-24), and start the training process (REQ-21). She is given the option to subscribe to process updates (REQ-17) and to monitor the training process (REQ-19). The backend receives the request, validates the configuration, creates and configures

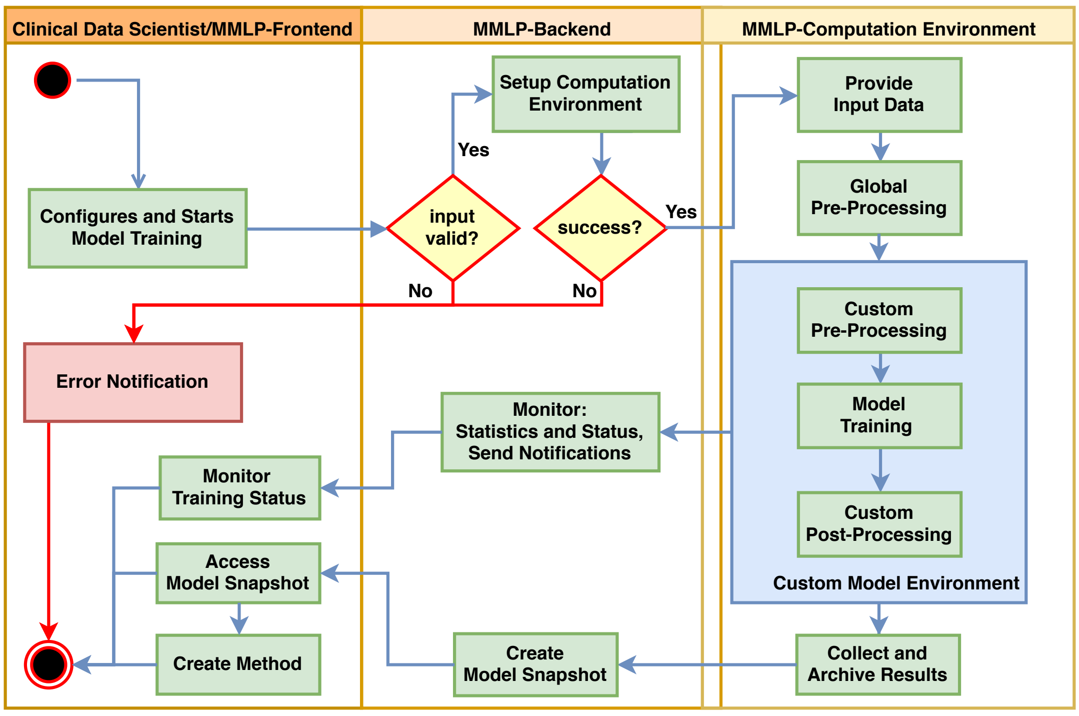


FIGURE 5.5: Activity diagram: Model training

the computation environment and provides the needed input data for the training (all needed input data are shown in Figure 5.7). The training is started and the pre-processing steps are applied, transforming the data set into a matching format for the model. Afterwards the model is trained using the provided data set and the configured hyperparameters. During post-processing stage the researcher can add tasks which help to analyse the training results. In general this step is rather optional during training, since training focuses on improving the model and not to produce annotations on unknown images. In the last step the model snapshot is created from the current state and the clinical data scientist can use the generated meta data to evaluate the performance of the newly created snapshot.

After the machine learning method training succeeded, the clinical data scientist is given the option to list (REQ-13), create (REQ-14), and remove machine learning methods (REQ-15). When the methods are updated, the medical expert is able to access and utilize the newly methods.

Method Application Process

The method application process is also largely depending on the computation environment. After the clinical data scientist created a method from her model snapshot, the medical expert can use it to analyze her data. Figure 5.6 shows how the medical expert and the platform communicates and which steps are required to apply a method to new data.

In the first step the medical expert selects a method (REQ-13) and uploads her patient data (REQ-16) using the frontend component. The backend validates the

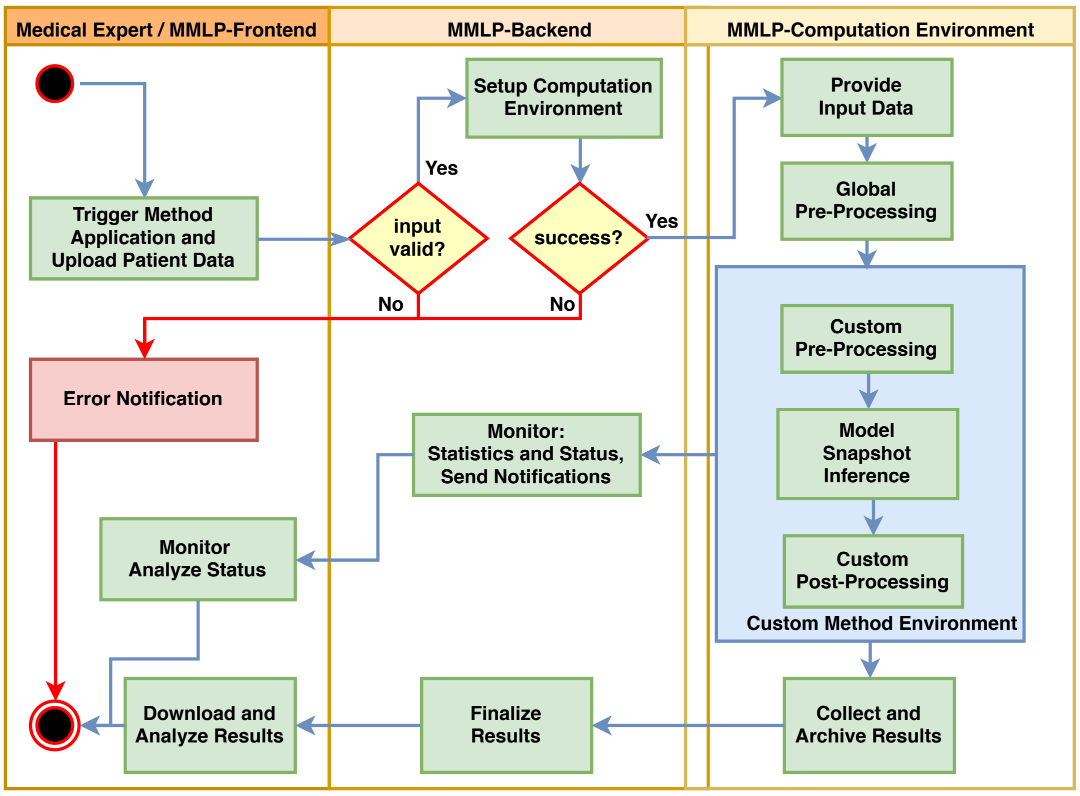

Medical Expert / MMLP-Frontend
MMLP-Backend
MMLP-Computation Environment
Trigger Method Application and Upload Patient Data
input valid?
Yes
No
Setup Computation Environment
success?
Yes
No
Provide Input Data
Global Pre-Processing
Custom Pre-Processing
Model Snapshot Inference
Custom Post-Processing
Custom Method Environment
Error Notification
Monitor: Statistics and Status, Send Notifications
Monitor Analyze Status
Download and Analyze Results
Finalize Results
Collect and Archive Results

FIGURE 5.6: Activity diagram: Method application

request, setups the computation environment, and provides the needed input data for the method (all needed input data are shown in Figure 5.8). The pre-defined pre-processing steps are applied on the uploaded patient data and the model snapshot is loaded to analyze the provided patient data (REQ-18). Afterwards the resulting annotations and the logs of the application generated and stored. All results are collected, compressed and archived. Notifications are send to inform the user about the current state of the analysis. To evaluate the results the clinical data scientist downloads the result-archive and views the annotations (REQ-20).

Standardized Computation Environment

The standardized computation environment has to support two use-cases: first the model training is described in Figure 5.7 and second the method application in Figure 5.8.

Before model training the data set is transformed into the required input format of the model. Additionally, further components could be added to manipulate the input data, such as image anonymization and augmentation. The resulting data is used to train the model for a specific tasks. Hyperparameters and other settings are provided to the model. If the model supports it, live tracking and monitoring services of the model are exposed to the user. This allows further possibilities to understand how the training performs and maybe ideas on how to improve it in the next iteration. After the training is finished, a custom-preprocessing pipeline allows to further process

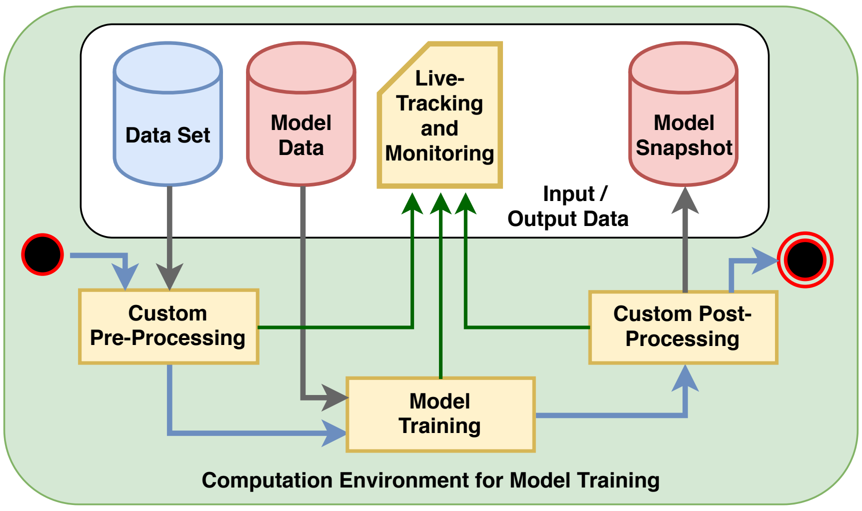


FIGURE 5.7: Standardized computation environment for model training

the data generated during training. The state of the model and generated files is preserved in a model snapshot. If the model snapshot is good enough, the clinical data scientist uses it to create a method.

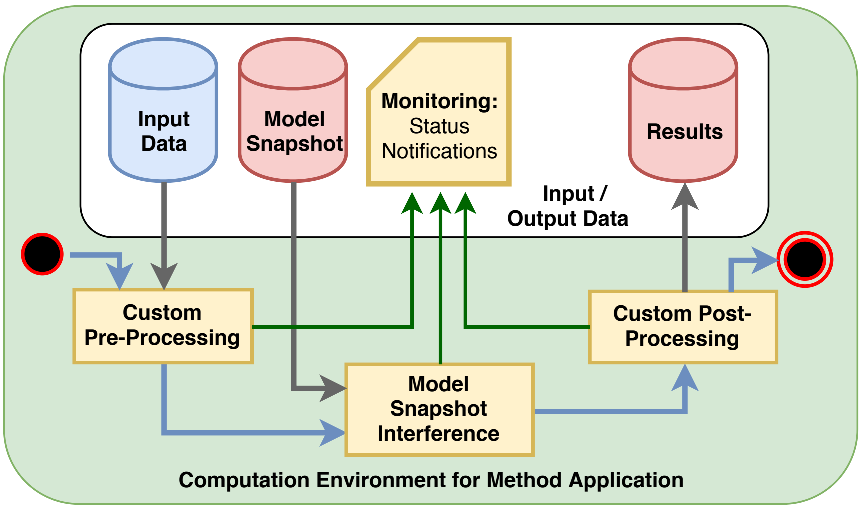

Input Data
Model Snapshot
Monitoring: Status Notifications
Results
Input / Output Data
Custom Pre-Processing
Custom Post-Processing
Model Snapshot Interference
Computation Environment for Method Application

FIGURE 5.8: Standardized computation environment for method application

The uploaded patient data must be pre-processed to analyze it using the machine learning method. Therefore the global and custom pre-processing steps are performed in the application pipeline. The method is applied and produces new annotations, such as showing the estimated location and severity of a tumor. In the last step the data and the annotations are post-processed to transform them into a standardized format, which has to be readable by the medical expert.

Computation Environment Implementation

For stability and reproducibility concerns, every time the model is trained or method is applied, the exact same environment should be present. Even small, unexpected changes in the filesystem or the installed software versions could cause serious issues. Therefore, the computing environment is implemented using virtualized application containers, specifically Docker, providing flexibility while having a negligible performance impact [32]. The computing environment requires specialized computing infrastructure providing multiple GPU's for training machine learning models. However, Docker does currently not support sharing of a GPU out of the box. To circumvent that issue, the platform relies on the Nvidia-Docker container runtime [25] for shared GPU access.

Another essential aspect is the flexibility of the environment to support different model frameworks. As the models' design and behavior can vary considerably depending on its intended purpose, a non-restricting standardization of the structure and execution routine can be challenging. Therefore, instead of standardization on the model-level, a standardization of the environment around that model is implemented in the platform.

Application containers allows the model-developer to install all kinds of requirements within the models' environment, such as specific versions of a library, and even allow custom builds of requirements. Thus, the clinical data scientist has a unrestricted pool of design choices when developing a model for the medical expert. If applicable, even non-machine learning can be implemented. In special cases, it is necessary to adapt the computation environment to the on-premises infrastructure. This includes local GPU clusters, a specific set of GPU workstations, or other kinds of specialized hardware. Additionally, it may be required to standardize the usage to only support a specific machine learning framework, to further slimline the process and offer exchangeability between the model and the components of the pre- and post-processing pipelines.

For this prototype, the generic application approach is implemented using Docker container. Docker supports the creation of virtual environments for nearly all kinds of programs. Compared to other solutions, it is especially lightweight, portable, and resource-efficient.

5.5 Infrastructure Stack

There are multiple options to provide the required compute resources, as described in Chapter 2. The simplest option is to install the platform components natively on top of the operating system. However, this does not scale well beyond one machine, other applications could interfere with the platform, and pre-installed libraries could cause version conflicts with the platform requirements. A possible solution is to leverage hardware virtualization and run the platform and its components within virtual machines. Virtual machines run a full operating system, which itself has a considerable performance cost. Our preferred solution is to use virtual application containers, which has similar advantages to virtual machines but only have a minor performance impact. Therefore the infrastructure stack has to support application containers and manage the available resources efficiently.

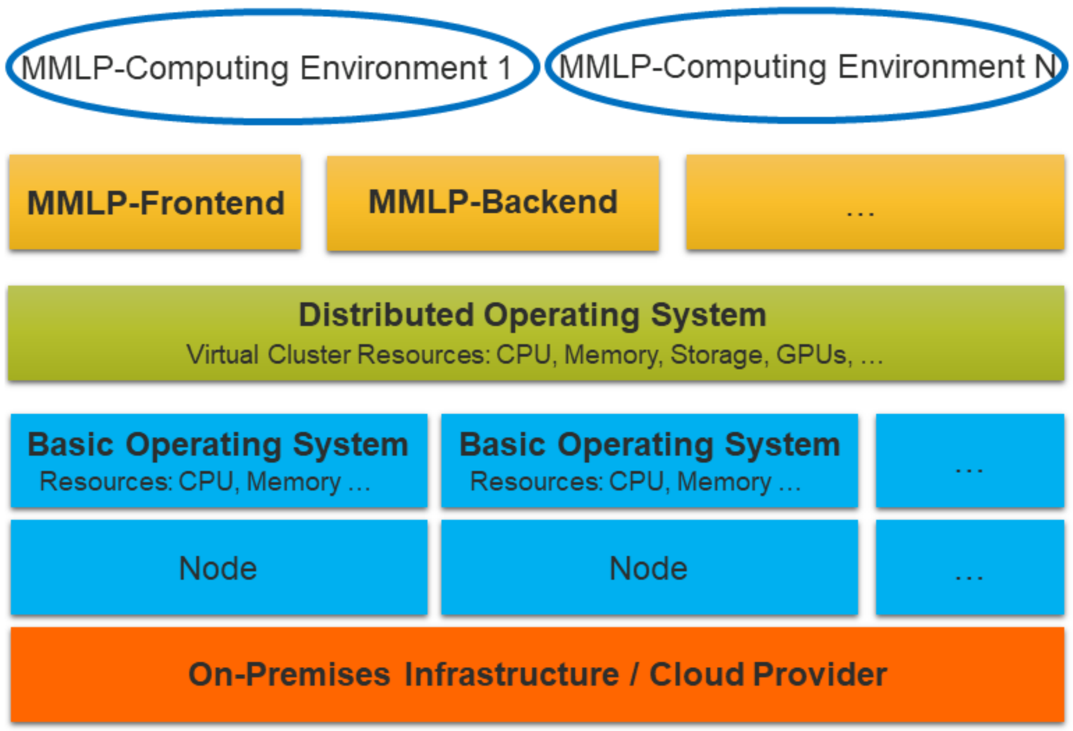

MMLP-Computing Environment 1
MMLP-Computing Environment N
MMLP-Frontend
MMLP-Backend
...
Distributed Operating System
Virtual Cluster Resources: CPU, Memory, Storage, GPUs, ...
Basic Operating System
Resources: CPU, Memory ...
Basic Operating System
Resources: CPU, Memory ...
...
Node
Node
...
On-Premises Infrastructure / Cloud Provider

FIGURE 5.9: Proposed infrastructure stack

An overview over the infrastructure stack is shown in Figure 5.9: The base layer represents the virtualized computing hardware stack, including the machines, networking and security rules, and monitoring. It is managed using automated tools, also called infrastructure as code (IaC) [68] service, or Infrastructure as a Service (IaaS). On top of the infrastructure, computing nodes are created and pre-installed with a lightweight, robust, and container-supporting operating system. The nodes are managed using a distributed cluster operating system. A distributed cluster operating system combines the resources of all available nodes, schedules new jobs, and monitors and controls the cluster. The developed Medical Machine Learning Platform is deployed on top of the distributed cluster operating system. Therefore the prototype assumes that the infrastructure is already deployed (open source templates that start the infrastructure are available [30], alternatively a managed service such as Amazon Elastic Container Service for Kubernetes [4])

Infrastructure Implementation

As motivated in Chapter 2, the base operating system CoreOS is used, due it's minimalistic, container-centered, and robust architecture [66]. On top of the base operating system, a distributed operating system is needed to make the resources of each node manageable and schedule the containers on the nodes. Kubernetes [13] is an open-source distributed cluster resource manager, introduced by Google in 2014. It fulfills our needs and is capable of executing our docker containers on the cluster.

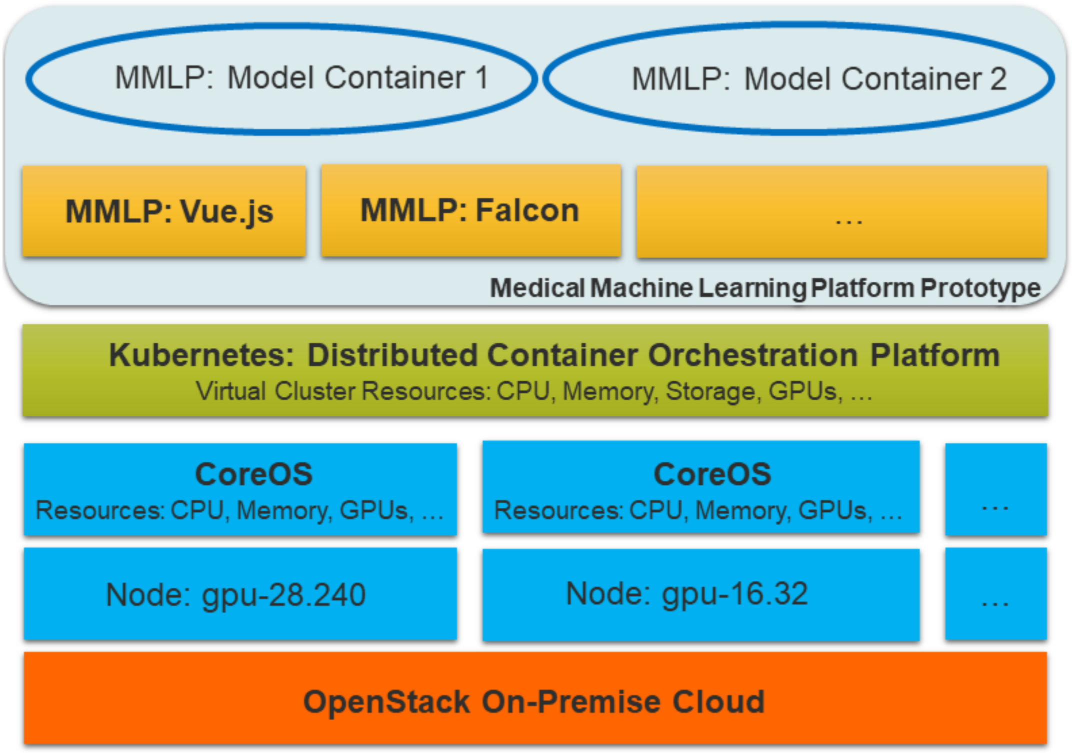

MMLP: Model Container 1
MMLP: Model Container 2
MMLP: Vue.js
MMLP: Falcon
...
Medical Machine Learning Platform Prototype
Kubernetes: Distributed Container Orchestration Platform
Virtual Cluster Resources: CPU, Memory, Storage, GPUs, ...
CoreOS
Resources: CPU, Memory, GPUs, ...
CoreOS
Resources: CPU, Memory, GPUs, ...
...
Node: gpu-28.240
Node: gpu-16.32
...
OpenStack On-Premise Cloud

FIGURE 5.10: Envisioned architecture of fully deployed platform

The fully deployed platform architecture is shown in Figure 5.10. The infrastructure consisting of an on-premise cloud provider, nodes, a basic operating system, and a distributed container orchestration platform (supporting Docker) is expected. On top of the infrastructure stack, the backend, frontend, and the computing environment are deployed. The backend is implemented using Falcon [35], a robust, fast, and minimalistic web framework. Model training and method inference is implemented in Docker [16] and supports all kinds of machine learning tasks (maybe even more generic tasks to be applied on images).

5.5.1 Platform Deployment

The platform is designed in a modular way, allowing simple deployments in every environment that supports application containers. The frontend and backend itself are deployed within a virtual application container each, to enable flexible and straightforward deployment without interference with other applications (Figure 5.11). Training and application pipelines are also running in their own application containers, allowing it to scale and run distributed across the cluster.

The here employed infrastructure could be integrated into local on-premise cloud solutions such as OpenStack [79], or public cloud solutions such as AWS [96], Google Compute Cloud[54], and Azure [94].

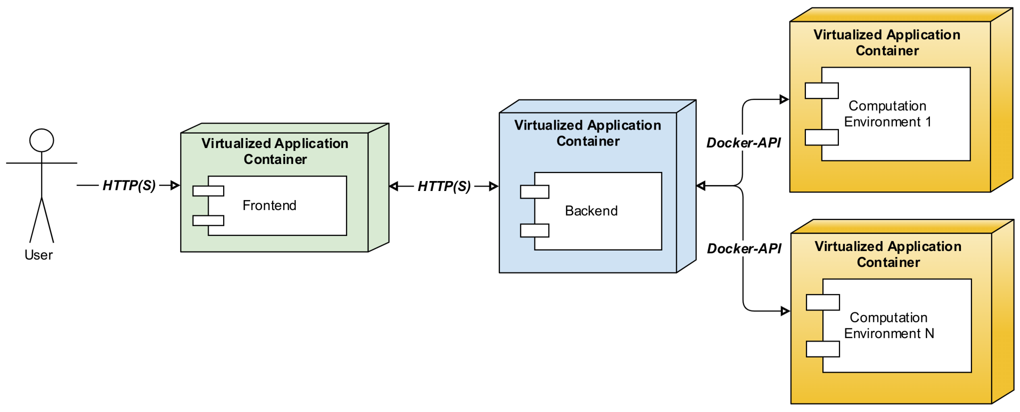

User
HTTP(S)
Virtualized Application Container
Frontend
HTTP(S)
Virtualized Application Container
Backend
Docker-API
Docker-API
Virtualized Application Container
Computation Environment 1
Virtualized Application Container
Computation Environment N

FIGURE 5.11: Platform deployment diagram

Chapter 6

Quality Assurance

This chapter provides answers to the following questions as described by Ludewig and Lichter [59]:

1. Are the system functions operating as expected?
2. How can the platform be developed in an understandable way to reduce the time new developers need to improve a specific aspect?

To prioritize the focus during testing and development, the following factors are considered: risk management and cost management. Risk management evaluates which component requires special attention to ensure that the most crucial platform functions are available. This includes especially the core functionality of the frontend, backend, and the computing environment. If one of them fails, the entire platform will not be reachable anymore. Other issues may affect a set of functions within the platform, but not all users might be affected. Therefore the core functionality is tested extensively. Cost management evaluates how much damage is caused if an error occurs in contrast to how much it costs to prevent it. However, this platform is a proof of concept and will not directly be used within a production setting. Therefore the focus is on minimizing the risk of failure.

Management of knowledge is an essential factor for the successful development of software. To meet the requirements for good software, especially the following points were considered during development: Functionalities are divided into small components to ensure the reusability for other platforms and new features within the platform, as shown in Figure 5.2 and Figure 5.4. The specification should clearly state which features are required, including all demands to the environment and the way the user interacts with it, as shown in Chapter 4. The documentation should include at least the description of each function, its parameters, how it is designed, and how it interacts with other components.

To improve the overall quality, three paths are recommended [59]: organizational, constructive, and analytics. The organizational path describes how the programming language is selected, the configuration is managed, the issues are handled and how they affect the system, and how the programming was managed, which is part of Chapter 5. The constructive path focuses on systematic development strategies, such as guidelines, templates, tools, methods and notation strategies and training. The analytic path validates whether the requirements are fulfilling by inspections from experts, static analysis using programming tools, and testing of the desired behavior. For this platform, the paths are adapted as follows:

- **Organizational Path**
  The programming languages were chosen specifically for the desired purpose of the component. Due to the containerization of the platform itself, the programming language is selected for each component independently, and thus the matching language is selected. Details of the selection process are described in Section 5.3. The configuration is centralized with the goal to define each setting only once to avoid inconsistencies and improve configurability. Functions with side-effects, meaning that the outcome of the function does not only depend on the input but also on other aspects such as reading files from the filesystem, are protected by try-catch statements. If something fails, the issue is logged, and the platform is ensured to have a clean recovery.

- **Constructive Path**
  Various parts of the platform require similar structures to perform. Templates are provided to guide the user to minimize the risk of failure. Additionally, the official code-style guidelines were followed as close as possible (e.g., PEP 8 [73] for Python and Lint for JavaScript [60]) and verified using the corresponding tools for development. These guidelines recommend specific notations to minimize various kinds of issues.

- **Analytic Path**
  During development, static code analysis was used in combination with a simple continuous integration pipeline. After changes to the codebase are made, the developer triggers an autonomous redeployment of all components and is able to validate immediately whether the desired code changes were successful.

6.1 Quality Assurance during Development

To test the platform prior to the evaluation, the usability, NFR, frontend, and backend tests were performed with potential users frequently during the development of the multiple prototypes. These tests focus on specific parts of the platform but also let the user space to explore and test further functionalities of the platform.

6.1.1 Useability-Test

Users were invited to test and evaluate to which degree the platform and its components are usable and intuitive. Users are presented real-world use case scenarios and are asked to solve them without detailed knowledge about the platform. The users of the platform are monitored while they try to solve the typical tasks with the platform. During this process, everything that could be helpful for development was noted. To improve the outcome, users are invited to apply techniques such as loud thinking, to allow the developer to understand how the platform can be improved and which functionalities might be missing. The test is executed as follows:

1. Prepare the functionality which should be tested

2. Prepare the platform for the test

3. Let users solve the relating tasks using the platform
4. Collect, discuss, and implement the feedback

6.1.2 NFR Tests

To measure some of the non-functional requirements, users are asked to answer a small set of questions about their platform-usage-experience. This includes multiple parts of the quality attributes, which are described in the standard for systems and software quality requirements and evaluation (SQuaRE). SQuaRE consists of two parts: quality in use and software product quality.

Quality in use is directly evaluated with the user and includes effectiveness, efficiency, satisfaction, freedom of risk, and context coverage.

The software product quality is validated during development and can partly be measured afterwards. The development was guided by the following key criteria:

- Functional suitability, which includes functional correctness (see Table 4.36), functional completeness (see Table 4.35), and functional appropriateness
- Performance efficiency, which includes time behavior (see Table 4.37), and resource utilization
- Compatibility with various types of data sets and models (see Table 4.34)
- Usability, which includes learnability, user error protection, and user interface aesthetics
- Reliability, which is tested while testing the other aspects
- Portability, which is part of the design using application containers

6.2 Backend Development and Testing

The quality of the backend is ensured by using a test-driven development [10] approach. Testing of the backend is performed utilizing the API development environment Postman [74].

The first step of implementing a new functionality of the backend is to define the API-Request, what it should do and which components are required to achieve this, which errors should be caught (e.g., false user input), how unknown errors are handled, and if it should run synchronously or asynchronously. This request is then defined in Postman. At this stage, the endpoint is not implemented, and the request will fail with a 404 not found error. From this point, the endpoint is improved iteratively, first by providing meaningful error messages and input validation checks and second by the functionality and all requirements to provide the requested functionality. If the functionality requires data objects (such as a machine learning model), mockup data is created and used. In the final step, real data, or close to real data (for sensitive data) is included.

These integration tests could be improved further by implementing tests on a lower level using python unit tests, checking every single function using equivalence partitioning testing. Equivalence partitioning testing ensures that all types of input values are tested and the outcome is either a meaningful error or the expected outcome.

6.2.1 Simple Static Code Analysis

To further measure the code quality, a static code analysis was performed. The shown code metrics were collected manually and by using the tool radon [92]. Additionally, to the fundamental metrics, the cyclomatic complexity is used as an indication for the complexity of the backend. The cyclomatic complexity is calculated according to [87], and the resulting score is ranked according to Table 6.1.

TABLE 6.1: QA: Cyclomatic complexity classes

Score	Rank (according to [87])
1 to 5	A (low risk — simple block)
6 to 10	B (low risk — well structured and stable block)
11 to 20	C (moderate risk — slightly complex block)
21 to 30	D (more than moderate risk — more complex block)
31 to 40	E (high risk — complex block, alarming)
41+	F (very high risk — error-prone, unstable block)

TABLE 6.2: QA: Backend code metrics

Metric	Measured Value
Number of Python Classes	54
Number of Python Functions	246
Lines of Code	4388
Average Cyclomatic Complexity	A (2.1)

The collected metrics are shown in Table 6.2. The complexity rank of A indicates good code quality, especially in terms of testing, maintainability, modularity, and reusability.

6.3 Frontend Development and Testing

The frontend is developed similar to the backend. First, the desired view for the user is defined. This includes how it should look, which and how the data should be shown, and all the required functionalities to archive this goal. Afterwards, the view, its components, the required functions, and the storage state management patterns are implemented. The communication with the backend to request real data can be very complex, especially if larger files are transmitted. This is typical for data sets, patient data, machine learning model snapshots, and results, which require asynchronous, multi-part messages, and a lot of fine-tuning to perform as expected. In the early stage of the development, mock data is utilized to guide the design process of a view. In later development stages, the mock data is replaced with real data to develop and test the full communication capability.

In the final step, the implemented view is compared to the previous definition of the view and tested against wrong user input, performance, and other non-functional requirements (see Section 4.9).

6.3.1 Simple Static Code Analysis

To further measure the code quality, a static code analysis was performed. The shown code metrics were collected manually, due to the lack of tools for static code analysis supporting Vue.js at the time of writing. Table 6.3 shows the collected metrics.

TABLE 6.3: QA: Frontend code metrics

Metric	Measured Value
Number of Functions	184
Number of JavaScript Functions	43
Number of Vue.js Functions	141
Lines of Code	5344
Lines of Code JavaScript	1661
Lines of Code Vue.js	3683

Chapter 7

Evaluation

This chapter describes how the evaluation of the platform was planned, performed, and discusses the results. The evaluation itself and the strategy behind it is described in Section 7.1. In Section 7.2, the results of the evaluation questionnaire are presented and discussed. The biases of this evaluation are discussed in Section 7.3.

7.1 Evaluation Strategy

The evaluation aims to validate how good the platform matches the requirements of the clinical data scientist and the medical expert. In particular, the evaluations focus on the ease of use, the collaboration support between both personas, and the possibility to use existing machine learning models. Therefore, both personae are confronted with the platform interface without further steps or instructions. This approach assesses the intuitiveness and the understandability of the user interface. The evaluation questions are summarized in Table 7.1.

TABLE 7.1: Governing evaluation questions

ID	Question
Eval-1	How good does the proposed platform match the requirements and expectations of clinical data scientist and medical experts?
Eval-2	Does the platform improve the collaboration between clinical data scientists and medical experts?
Eval-3	How can the platform be further improved?

Before each evaluation, the platform state is reset, to ensure that every participant evaluates precisely the same platform, especially with the same data sets, models, model snapshots, methods, and results.

The evaluation is performed following the questionnaires, which are shown in Appendix A. The design focusses on user acceptance, leveraging the technology acceptance model (TAM) [89]. TAM primarily focusses on the usage without prior training to measure the perceived usefulness and the perceived ease of use.

To ensure meaningful results and further investigate the usability and intuitiveness, evaluators are required to think aloud during all parts of the evaluation, especially during the parts where they perform tasks using the platform. During the evaluation, the participant's actions and approaches are monitored, to gain further insides on how the platform could be further improved. In the end, a brief discussion with the participant was offered, to provide further information and collect additional feedback about the platform itself and further application possibilities.

7.1.1 Personae

In the first part, details and descriptive data about the evaluating person and their work environment are gathered. This includes the amount of experience in their field, their tasks, whether they cooperated with others before, and whether they see the potential for closer collaboration.

Furthermore, the biggest hurdles for applying machine learning methods in the medical field are evaluated. The assumed hurdles include awareness of the currently available state-of-the-art methods, simple access to them, the computing capabilities, establishing the machine learning workflows, and the lack of collaborations to improve the available methods.

7.1.2 Platform Evaluation

In this paragraph, the evaluators are asked to perform their main user tasks such as training machine learning models or analyzing patient data. To further evaluate the design for the medical experts, the clinical data scientists are also asked whether they find the abstraction of the analysis/inference process of the medical experts views is clear and sufficient. The platform provides specialized frontends for each of the two personae. Therefore, the workflow for each personae is evaluated as follows:

Platform Evaluation Workflow for the Clinical Data Scientist

The clinical data scientists main goal is to use the platform to develop new machine learning models. During evaluation the main steps are performed by multiple clinical data scientists. These steps are described in the following list:

1. Import a provided data set (containing image and segmentation objects)
   System:
   The user-defined data set is requested and retrieved by the system.
   Result: The user has access to the newly imported data set.

2. Import a Machine Learning Model into the Platform
   System:
   The platform validates that the provided source-url is valid, requests a copy of the models source code, and imports it to the platform.
   Result: The user has access to the model and its versions within the platform.

3. Fine-Tune a pre-trained model snapshot, using the imported data set, and improved hyperparameter settings. The platform guides the user through each step of the training configuration and allows to select the previously trained model snapshot.
   Required User Actions:

   (a) The user selects the data set

   (b) The user selects the machine learning model

   (c) The user selects a previously trained model snapshot for training.
   (d) The user specifies two hyper parameters for the model (hyper parameters are model dependent and only shown as example). The platform automatically subscribes the user for notifications on a previously defined e-mail address.
       - **Number of Epochs** should be set to 1
       - **Preprocessing** should be active.

       System:
       The platform validates the input of the user, the data set, the machine learning model, and the selected training configuration. It then creates a new model snapshot and documents the metadata and provided settings. The running training pipelines and their statuses are shown to the user on the snapshot page. The platform sends important update notifications directly to the user via e-mail.

   Result: The user has access to the improved model snapshot. The user can download the metadata and the results (statistics, training history, trained weights) of the model snapshot. Additionally, the user received an e-mail containing important information about the pipeline, including the output of the pre-processing, the model during training. The actual results are assumed as successful since the model itself is not the focus of this evaluation.

4. Create a machine learning method using the trained model snapshot. After the training pipeline finished, the corresponding model snapshot is used to create a new machine learning method.
   Required User Actions:
   (a) The user lists available model snapshots of the model
   (b) The user creates a new machine learning method based on the newest model snapshot
5. The clinical data scientist switches to the interface of the medical expert and performs the analysis of provided patient data.
   Required User Actions:
   (a) The user switches to the interface for medical experts
   (b) The user lists available model snapshots of the model
   (c) The user creates a new machine learning method based on the newest model snapshot
   (d) The user lists and downloads the results

Platform Evaluation Workflow for the Medical Expert

The medical expert leverages the platform to analyze patient data using provided machine learning methods. During evaluation the primary workflow is performed by multiple medical experts. These steps are described in the following list:

1. Check the available machine learning methods
   Required User Actions: The user navigates to the method overview page of the platform.

2. Application of a machine learning method to the provided patient data. The idea is that the methods can be used with previously unknown patient data. This is one of the primary purposes of this platform since the application of methods can support *medical experts* in extracting valuable information from patient data.

   Required User Actions: The user navigates to the analysis page of the platform, and the platform guides her through the following steps:

   (a) The user uploads a patient data archive
   (b) The user selects the machine learning method
   (c) The user starts the analysis processing pipeline **System:**
   The system deploys a new analysis process and documents the metadata and provided patient data. The running processes are shown to the user on the dashboard. **Result:** The user can see and download the results.

3. Viewing and downloading of analysis results. After a new model was trained and exported to a method, it may be interesting for the user to check how the method performed on provided patient data.
   Required User Actions:
   The user navigates to the results page, and downloads the matching result.
   Result: The user can view and download the results of the applied method.

7.1.3 Design and Workflow Evaluation

One goal of the platform was to support as much as possible types of data sets and machine learning models. Therefore the clinical data scientist should be asked if it would be possible to integrate their machine learning model, including the required data sets and pre- and postprocessing into the platform.

After the evaluators have used the platform, they should evaluate their experience. The questions should cover the following topics:

- Overall impression of the platform
- How good the frontend guided the user through the complex workflows
- How responsive the platform reacted
- The aesthetics of the interface

The last question evaluates whether the evaluator sees that the platform contributes to a closer collaboration between both personas, which is a primary goal of the platform.

7.1.4 Additional Remarks

In the last step, additional remarks such as comments, observations, and concrete examples are collected to evaluate how easy or difficult it was to do a specific task. It should be clear that critics are very welcome to motivate the evaluators further to write about all the impressions they had during the usage of the platform.

7.2 Results and Discussion

This section describes the findings of the evaluation based on the questionnaire, loud thinking, monitoring the participants actions and approaches, and a brief discussion with the participant in the end. To maximize the outcome of the evaluation, each participant was evaluating the platform sequentially.

The evaluation was performed with 22 participants, consisting of 12 clinical data scientists and 10 medical experts. Before the participant arrived, the platform was reset to the same primary state. Questionnaires did not collect any personal information but do have a unique identifier. A separate documentation sheet for each persona collects each identifier of the questionnaire and the signing of the participant. For privacy concerns, each participant was only able to see his identifier and name, and the others were hidden.

7.2.1 Clinical Data Scientist

The clinical data scientist evaluates the platform, especially focussing on the machine learning model development. On average, the evaluation took 40 to 60 minutes, including discussion of further feedback and questions.

Persona

The average clinical data scientist has about three years experience in the medical field (the inexperienced participant with a half year, and the most experienced ten years). 58% of the participants worked with medical experts before, and all participants see the potential for closer collaborations.

Most of the participants are working on multiple tasks such as image segmentation (N=7), object detection (N=4), regression (N=3), image registration (N=3), classification (N=3), and seven other tasks. Clinical data scientists see the most significant hurdles when applying machine learning models in the effort to establish the required machine learning workflows (N=10), awareness of existing and performance of machine learning methods (N=8), lack of collaborations (N=6), simple access to state-of-the-art machine learning methods (N=6), lacking computing capabilities (N=6), and four other issues. Among the other issues, the following were mentioned: the difficulty of reproducing results on the own data, lacking integration to the medical workflow, and dependency issues of the code.

Platform Evaluation Results

The participants were asked to perform a generalized workflow containing the required steps to fine-tune a pre-trained model within the platform. The results are shown in Figure 7.1.

In the first step the participants had to find out how a medical data set intuitively can be uploaded (*S1*), how a model can be imported from an external repository (*S2*), how a pre-trained model can be fine-tuned and customized using the training pipeline (*S3*), how the newly created model snapshot can be viewed and downloaded (*S4*), and how the model snapshot can be exported as a method (*S5*) to support medical experts with their workload.

In *S6*, the clinical data scientist evaluates, how the abstraction of the interface and the workflow for the medical experts is designed. *S7* and *S8* aim to check the possibility to perform the clinical data scientists research within the proposed platform.

The main goals of the platform are to improve the model development task and support the collaboration between the personas by reducing technical barriers. *S9* evaluates how good these goals were archived.

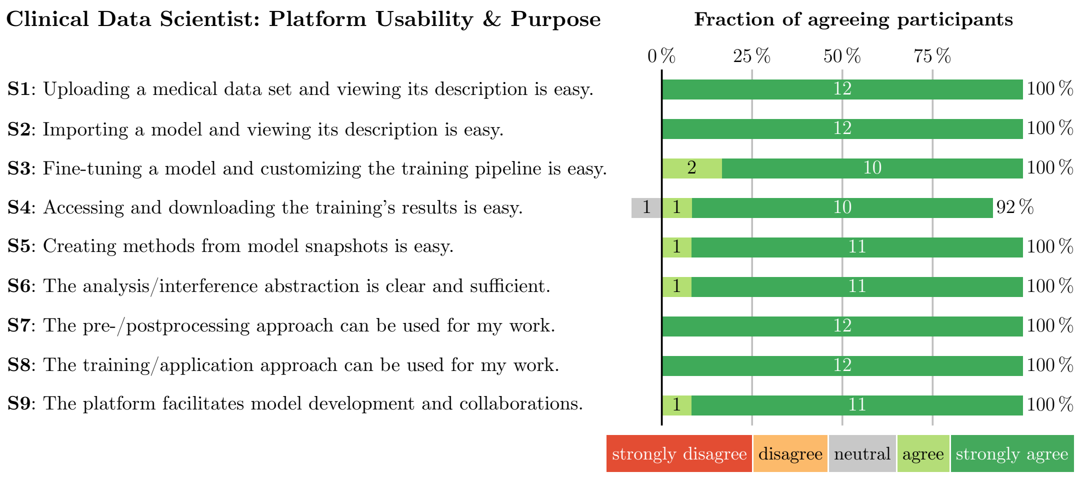


FIGURE 7.1: Clinical data scientist: Evaluation platform useability and purpose

The results are shown in Figure 7.1 show that the 12 participating clinical data scientists agreed in all points by a minimum of 92%, with one neutral answer.

Overall Platform and Workflow Design

After the participants performed the fine-tuning and hyper parameter customizing of the provided model on the uploaded data set, the overall platform experience is focussed. The following four aspects are addressed to evaluate the overall platform experience: overall impression, guidance and workflows, aesthetics, and responsiveness.

The results are shown in Figure 7.2. The overall impression was rated very good (5) by 92% of the participants (mean: 4.9), 67% rated the guidance and workflows provided by the platform as adequate guidance and clear instructions (mean: 4.7),

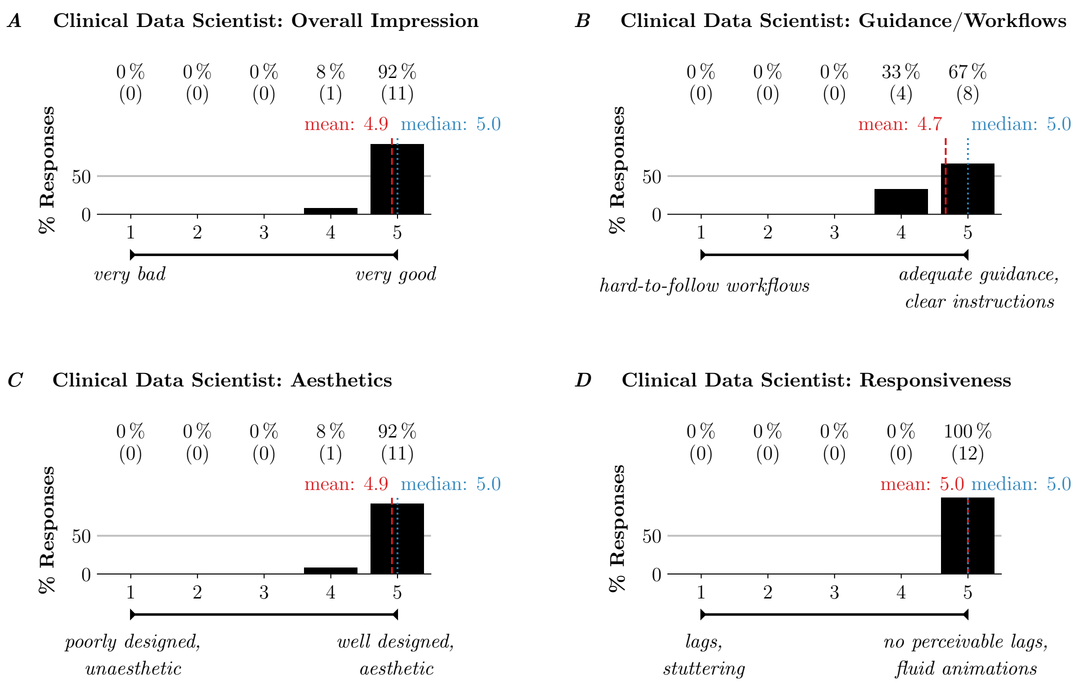


FIGURE 7.2: Clinical data scientist: Evaluation platform overall rating

92% rated the aesthetics as well designed (mean: 4.9), and no participants did find any perceivable lags (mean: 5).

Additional Feedback

In the last step, each participant had the chance to provide additional feedback for the platform. In this scope, only frequently mentioned feedback is presented, the complete feedback is shown in Section B.2. Nine clinical data scientists suggested to integrate the platform with broader, on-premise compute resources, four would expose more functionality for the data sets, models, training pipeline, and methods to the frontend, and three would integrate further automation tools to the training pipeline such as hyperparameter optimization.

Discussion Clinical Data Scientist

Clinical data scientists are using the platform to manage data sets, manage machine learning models, train machine learning models with the data sets, sharing trained, pre-configured models as methods with medical experts, and managing the results. The platform removes a lot of technical barriers by offering high automation and abstraction of sophisticated features, without restricting the user too much.

By focussing on the workflow to iteratively improve machine learning models and collaborating with the medical expert for development and application, it was possible to design the user interface very intuitively but still flexible enough to customize the training configuration and hyperparameters. The current prototype builds on the experience we collected from two prior developed prototypes, especially the backend functionality and frontend design.

During evaluation, 12 clinical data scientists with half to 10 years of experience confirmed significant technical barriers to applying machine learning methods on patient data of the medical experts. Fortunately, they agreed that the proposed platform eases these barriers. When asked if it would be possible to integrate their current machine learning developments, including all required pre- and post-processing components, training- and application pipeline, they strongly agreed that the platform could be used. Especially for the collaboration with medical experts, clinical data scientists agree that the platform eases the technical barriers. Besides, the overall impression of the platform, guidance, and aesthetics of the user interface was very well perceived. There were also additional questions about the scalability of the compute resources, especially using already operating GPU-clusters and the possibilities to integrate further advanced tools to perform tasks such as automated hyperparameter optimization.

These features would integrate nicely with the platform. Integrating on-premise computing clusters require the support of local administrators and an additional option for the user to select the location for the computation environment. Additionally, other computation environments could be used to respect limitations such as data location, licensing, and infrastructure cost. Currently, Docker is supported for training and inference of models. The computation environment management could be extended to also support other technics, such as Kubernetes Pods [13], and managed (cloud) services such as the Amazon Elastic Container Service [3].

Advanced features, such as customized pre- and post-processing, augmentation, and automated hyperparameter optimization are benefiting strongly from the modular design of the platform. These tools can be added to the matching component of the platform, without changing the other parts. For example, hyperparameter optimization approaches could be integrated as the first step after the user committed the training configuration, and before the training is actually started. Since the whole training is now parametrized, the hyperparameter optimization tool is able to start multiple training iterations, modifying the hyperparameter settings as required to improve the model's performance.

7.2.2 Medical Expert

The medical expert evaluates the platform, especially focussing on the analysis of local patient data using provided machine learning methods. On average, the evaluation took 20 to 40 minutes, including discussion of further feedback and questions.

Persona

The average medical expert has six years experience in the medical field (the inexperienced participant with two years, and the most experienced 15 years). 60% of the participants worked with clinical data scientists before, and all participants see the potential for closer collaborations and potential to apply machine learning methods in their field.

Most of the participants are working on multiple tasks such as image segmentation (N=6), image registration (N=4), image reconstruction (N=2), and 12 other tasks. Medical experts see the most significant hurdles when applying machine learning methods in the effort to establish the required machine learning workflows (N=8), lack of collaborations (N=8), lacking computing capabilities (N=7), simple access to state-of-the-art machine learning methods (N=6), awareness of existing and performance machine learning methods (N=5), and four other issues. Among the other issues, the following were mentioned: the difficulty of reproducing results on the own patient data, and the problematic access to clinical data.

Platform Evaluation Results

The participants were asked to perform a generalized workflow containing the required steps to view the available machine learning methods and to analyze their uploaded patient data using a machine learning method within the platform. The results are shown in Figure 7.3.

In the first step, the participants had to intuitively find out how they can view all available machine learning methods within the platform (*S1*), how to use the segmentation method to analyze locally available patient data (*S2*), and how the results of the analysis can be accessed and downloaded (*S3*).

The main goals of the platform are to improve the machine learning method development process and to support the collaboration between both personas by reducing technical barriers. *S4* evaluates how good these goals were archived.

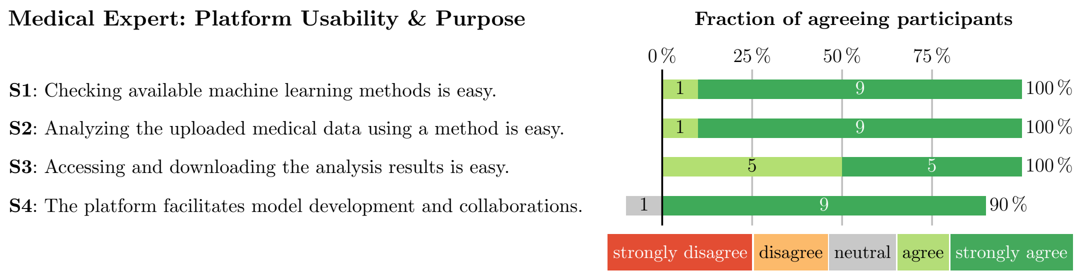


FIGURE 7.3: Medical expert: Evaluation platform useability and purpose

The results in Figure 7.3 shows that the ten participating medical experts agreed in all points by a minimum of 90%, with one neutral answer for S4.

Overall Platform and Workflow Design

After the participants performed the analysis of the provided patient data using the segmentation method, the overall platform experience is focussed. The following four aspects are addressed to evaluate the overall platform experience: overall impression, guidance and workflows, aesthetics, and responsiveness.

The results are shown in Figure 7.4. The overall impression was rated very good (5) by 90% of the participants (mean: 4.9), 90% rated the guidance and workflows provided by the platform as adequate guidance and clear instructions (mean: 4.9), 80% rated the aesthetics as well designed (mean: 4.8), and no participants did find any perceivable lags (mean: 5).

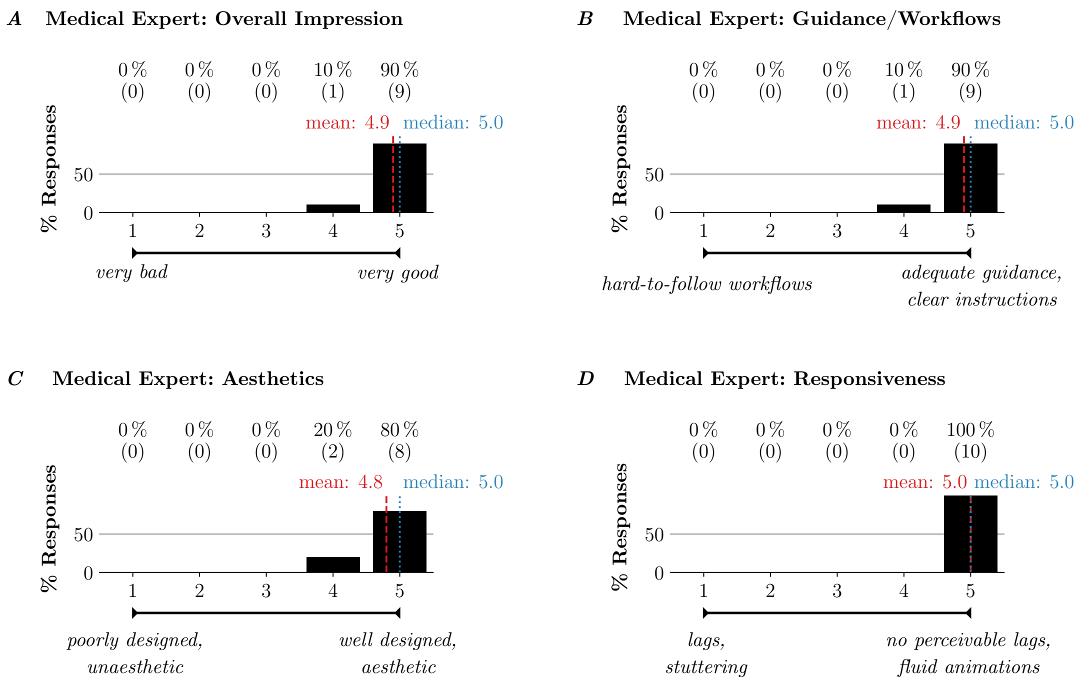


FIGURE 7.4: Medical expert: Evaluation platform overall rating

Additional Feedback

In the last step, each participant had the chance to provide additional feedback for the platform. In this scope, only frequently mentioned feedback is presented, the complete feedback is shown in Section B.1. Four medical experts suggest adding an option to reduce the debug information in the analyze results, to further simplify the results. Another point is that four medical experts are interested in developing machine learning methods themselves, but faced lots of restrictions in terms of complexity, lacking compute resources, and knowledge on how to set up the required environment to do so. They mentioned that the platform strongly reduces the barriers for them to start developing new methods. Additionally, three medical experts see significant potential if this platform would be available to them in the future.

Discussion Medical Experts

Medical experts are using the platform to access the developed machine learning methods within the platform. The platform removes a lot of technical barriers by offering high automation and abstraction of sophisticated features, without restricting the user too much. By focussing on the workflow to uploaded and analyze patient data using machine learning methods, it is possible to radically slimline the interaction and knowledge required by the medical expert. To develop and customize the methods to the medical experts' needs, the clinical data scientists can use various types of medical data and machine learning frameworks.

During evaluation, ten medical experts with two to 15 years of experience confirmed significant technical barriers to applying machine learning methods on their patient data. Fortunately, they agreed that the proposed platform eases these barriers. Besides, the overall impression of the platform, guidance, and aesthetics of the user interface was very well perceived. They were also additional questions about when and where the platform will be available in the future, as well as if it would be possible for a medical expert to switch to the clinical data scientists interface to start learning how to develop and improve new machine learning methods. The platform would remove lots of barriers of developing new methods, since the medical experts are able to use existing methods and learn how to customize them to their needs without having to buy, configure, and set up specialized hardware, learn the required programming language(s), and set up a development environment. Within the platform they can directly modify the configuration and hyperparameter of a model, allowing them to start working on the machine learning research field, with a very steep learning curve and focussing on the training itself rather than all the other tasks around it.

7.3 Threats to Validity

Multiple biases could influence the outcome of the evaluation. To analyze these biases, the internal and external factors are discussed [98]:

Internal validity focuses on questions about how the evaluation questions related to other variables such as the previous experience of the participants. Therefore the questionnaire includes questions about the persona such as previous experience and work topics, which could be used to find correlations between the answers and the personas. Additionally, some of the participants know the author of this paper, which could influence their opinion during evaluation.

External validity focuses on the generalizability of the results. During evaluation, a pre-defined and configured model was used. Adapting and importing new models and their requirements into the platform could require more work and uncover areas to improve. Due to the limited time with the platform, participants may not have noticed all parts of the platform. After working with the platform more intensively, more feedback could be provided.

Chapter 8

Conclusion and Outlook

8.1 Conclusion

It is a challenge to translate the latest advances in academic research fields such as computer science and physics into clinical practice. A close collaboration between data scientists and medical experts is required to steer the translational process. In machine learning-assisted medical research, considerable effort is invested in developing and optimizing machine learning models. Exemplary applications are the automated segmentation of diseased tissue in MRI or CT scans, or the analysis of time-series data, such as ECG and EEG. Challenges include the accessibility, usability, and compute requirements of state-of-the-art machine learning research. As these challenges are not of algorithmic nature and therefore only indirectly related to the research task at hand, they hamper the impact of newly developed machine learning methods. In this thesis, we aimed to ease the use of state-of-the-art machine learning models and establish a machine learning platform that serves as a collaboration hub for medical experts and clinical data scientists. To achieve this, we systematically investigated three research questions (see Chapter 1.1):

First, a systematic literature research examined current state-of-the-art and state-of-the-practice to create medical machine learning platforms, the management of machine learning models, and the version management of large data sets (see Chapter 3). We found that there are multiple approaches available, but to our understanding, none of them offers the required flexibility for different machine learning frameworks, different types of data sets, and support for close collaborations between the medical expert and the clinical data scientists.

Second, we conducted a requirement analysis for a new machine learning platform (see Chapter 4), formulated a conceptual design and infrastructure stack (see Chapter 5), and implemented a realization thereof in a fully functional prototype, following the design goals defined in Table 1.2. The prototype is built on top of a scalable infrastructure stack, hosting three fully-virtualized – and therefore scalable – components: a web-based user interface, a python-based backend, and a standardized computation environment. Clinical data scientists can utilize and customize a comprehensive feature set to define, train, and share machine learning methods, including pre- and post-processing modules. On the other hand, the platform offers medical experts an easy-to-use workflow to import local patient data and subsequently apply pre-defined machine learning methods to it.

Further, the implementation of the platform was guided and tested by following an organizational, constructive, and analytic perspective of quality assurance (see

Chapter 6). The tests indicated the fulfillment of functional and non-functional requirements, as well as good code quality, which can be related to adhering to best practices in code testing, maintainability, and modularity.

Third, we performed and discussed a throughout evaluation of the platform prototype with both medical experts and clinical data scientists (Chapter 7). Both personae almost unanimously rated their impression of the platform as *very good*, stating that the offered feature set is easy-to-use, well-designed, and beneficial for collaboration between clinical data scientists and medical experts. The results of the evaluation make a strong case in favour of our final design and implementation decision. Further, they highlight the high interest in the proposed machine learning platform.

The conducted expert survey suggests that the here proposed platform could be used to develop, optimize, and apply machine learning methods in the medical domain and beyond. The platform is especially advantageous because of its ease-of-use, flexibility, and by-design scalability through virtualization and cloud-native approaches. Virtualization offers an almost unrestricted choice of machine learning frameworks and pre- and post-processing modules. Furthermore, the proposed abstraction of training, pre- and post-processing, and packaging it together into machine learning methods, could additionally motivate the acceptance of state-of-the-art machine learning in daily clinical use.

8.2 Outlook

Multi-Institute-Workflow The platform provides a generic workflow, which allows them to collaborate with multiple users on developing new machine learning methods. A similar concept could be applied to collaborate with multiple medical research institutions. One significant difference is the authentication of the users to access the available data sets, models, and methods. Additionally, data protection is much more difficult between multiple institutes, because after training a model on a sensitive data set, the model could be sensitive, too. Also, the communication channels could be provided by the platform, to collect relevant information and inform affected users, e.g., a request for a new method was made, which results in notifications for the interested clinical data scientists. Figure 8.1 shows three medical institutions, one focussed on patient-care and two focussed on machine learning in the medical context. The host of the platform is making their data sets, models, and methods available to other institutions they provide access to. Therefore the others are able to use high-quality data sets if they add their machine learning model to the platform. On the other side, researchers can collaborate on specific projects, to further improve the methods used in the medical field. The medical institute can access the developed methods, which allow them to leverage state-of-the-art machine learning techniques on their patient data.

Advanced Decision Support for Machine Learning During configuration of the training pipeline, the developer has to make lots of decisions to adjust the parameters. Knowledge collected during previous training pipelines could support a user decision support feature. For each parameter, the system could provide a suggestion to support

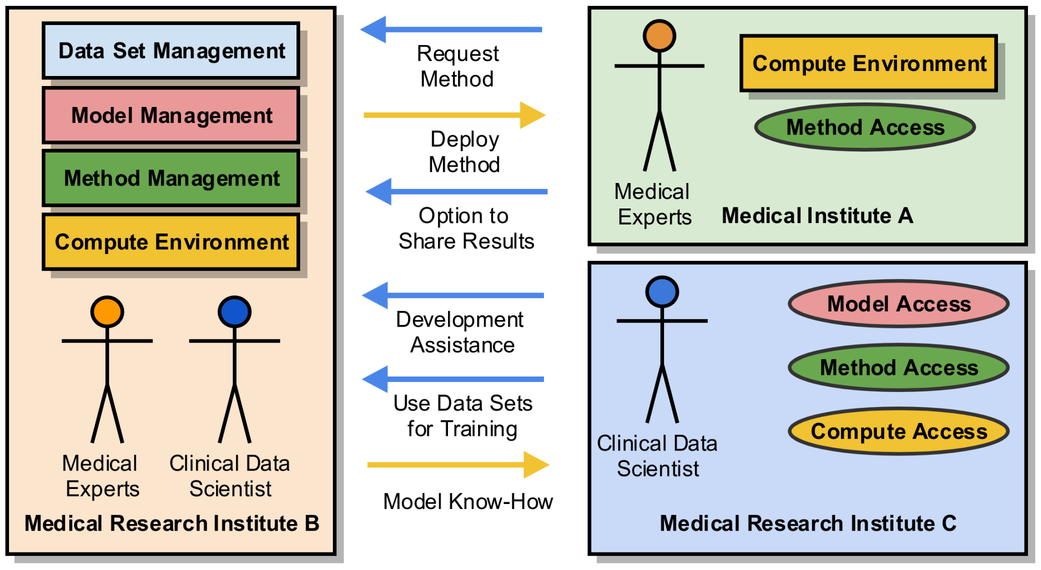


FIGURE 8.1: Multi-institute method development life-cycle

the developer. Additionally, all parameter values could be checked before a training pipeline is started to inform the user about possible issues or inefficient settings.

Edit the Models Source Code within the Web-Interface Developing new machine learning models sometimes requires to adjust parts of the source code. At some point, this could eventually allow the developer to work only within the web-interface, without requiring to set up a local machine learning capable workstation. Currently, the platform can load the changes from the model repository but does not support adjustments directly.

By adding a web-based text editor, such as a Jupyter notebook [45] the developer could access and modify the source code directly. The challenging part for this feature is the integration with the version control system. Changes of the user must be stored as a new model version or reverted to ensure that the source code of the model is not modified in an uncontrolled manner.

Advanced Statistics of a Model Snapshot Training a machine learning model may require multiple iterations. After each iteration, the results of the training have to be evaluated. Therefore the platform currently sends collected logs via email and offers a download of the model snapshot. Additionally, it could be beneficial to include vital statistics and graphs in the web-interface. The developer could check the results in a much simpler way and would not need to download and analyze the results manually.

Compare Advanced Statistics of Multiple Model Snapshots When statistics of model snapshots are available in the web-interface, it could be useful to compare multiple model snapshots. Especially interesting could be the influence of the provided model hyper parameters or data sets. The comparison could, therefore, show the differences in the configuration of multiple model snapshots and advanced statistics of the training.

Frequently used Pre-, Post-Processing, and Augmentation Container Management After importing data sets into the platform, it could be required to pre- or post-process them when used for model training. Therefore the platform could provide an overview of available containers for the developers to choose from. Additionally, multiple containers could be combined with a pipeline, to allow more complex processing steps.

Dynamic, On-Demand Scaling of Resources The amount of resources provisioned for a platform is usually determined before the installation of its software. There are multiple strategies to estimate the usage and calculate the amount of required resources, capable of handling the average and adjusting the infrastructure accordingly. Instead of estimating the resource demand, the resources could scale automatically to the required amount. This approach is especially common in cloud environments, and could also have great potential in this platform. If multiple users use the platform simultaneously, each training pipeline could require one or more GPUs and a large number of CPU cores and memory. Dynamic scaling could provide the required resources to a degree, that for each launched training or application, new resources are requested and released afterward. In terms of pricing, this approach allows to only pay for actually needed resources but could cause higher costs if the usage is not controlled or restricted.

Dynamic Computation Environment Location Deployment Depending on factors such as the used data set, license, model, and the required resources, it could be beneficial to use different compute environment locations. These locations could be selected automatically by the system or provided by the developer during configuration of the training pipeline. As an example, if a published model should be improved using publicly available data sets, it could be useful to use a public cloud provider, which usually can provide a vast number of resources. On the contrary, if sensitive data is used, it might be better to keep it within a protected computing environment, where resources are usually more limited and costly. The compute environments locations could include the local compute infrastructure, on-premise cloud providers, and public cloud providers.

Appendix A

Appendix: Evaluation Questionnaires

Questionnaire about the Medical Imaging Platform (MIP) for the Medical Expert

Persona:

I have been working in the medical field since approx. ☐ years.

Please describe the tasks you wish to use new machine learning methods for:

I have worked with clinical data scientists to develop machine learning models:	☐ yes	☐ no
Do you see the potential for the application of machine learning methods in your field?	☐ yes	☐ no
I see potential for closer cooperation with clinical data scientists in my work:	☐ yes	☐ no

Where do you see the biggest hurdles to apply machine learning models?
(multiple selections possible)

☐ easy/simple access to (state-of-the-art) machine learning methods

☐ awareness of the existence/performance of machine learning solutions

☐ computing capabilities

☐ establishing machine learning workflows is too time-consuming

☐ lack of collaborations

☐ other: ☐

Platform Evaluation:

1. **Verify** that a segmentation method is available within the platform.

   Viewing available methods is easy.

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

2. **Analyze** the provided patient cohort using the segmentation method.
   The platform guides you to the required steps. The patient cohort is provided as “patient_cohort.tar.gz”.

   Analyzing the provided patient cohort was easy

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

3. **View and Download** the analysis results.
   The platform sends a notification email after the analysis is finished.

   Viewing and downloading the results of the method application is easy.

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

Questionnaire about the Medical Imaging Platform (MIP) for the Medical Expert

Design & Workflow Evaluation:

Overall impression:

very bad ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 very good

Guidance/Workflows:

hard-to-follow workflows ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 adequate guidance, clear instructions

Responsiveness:

lags, stuttering ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 no perceivable lags, fluid animations

Aesthetics:

poorly designed, unaesthetic ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 well designed, aesthetic

Purpose:
The proposed platform could ease the deployment of machine learning models and could, therefore, contribute to a closer collaboration between medical experts and clinical data scientists.

☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

Additional remarks:

Please add further comments, observations, or other feedback in the box below. If applicable, please add concrete examples to your comments. General thoughts and critics are very welcome: What is easy and why is it easy? What is difficult and why is it difficult?

Questionnaire about the Medical Imaging Platform (MIP) for Clinical Data Scientists

Persona:

I have been developing machine learning models since approx. ☐ years.

Please describe the tasks you wish to develop machine learning models for:

☐

I have worked with medical experts to develop machine learning models: ☐ yes ☐ no

I see potential for closer cooperation with medical experts in my work: ☐ yes ☐ no

Where do you see the biggest hurdles for medical experts to apply machine learning models?
(multiple selections possible)

☐ easy/simple access to (state-of-the-art) machine learning methods

☐ awareness of the existence/performance of machine learning solutions in the expert's field

☐ computing capabilities

☐ establishing machine learning workflows is too time-consuming

☐ lack of collaborations

☐ other: ☐

Platform Evaluation:

1. **Upload** the provided "Hippocampus.tar.gz" data set from the default directory and verify that it is present.

   Uploading and checking available data sets is easy.

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

2. **Import** the machine learning model from *"git@github.com:magreiner/unet-docker.git"* and inspect its description.

   Importing models and viewing their description is easy.

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

3. **Train** the model "*U-Net Segmentation Model*" (newest version) on the previously uploaded data set using the pre-trained model snapshot. During the parameter customization dialog locate the pre-processing option and ensure that "*Medical Decathlon Preprocessing Container*" is set to ''true''. Additionally, adjust the hyperparameter "*Number of Epochs*" to 1. The platform guides you through the subsequent training configuration steps. The platform will save the resulting trained model as a "model snapshot".

   Training a model and customizing the configuration (pre-processing, hyperparameter selection, post-processing, …) is easy.

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

4. **View and Download** the newly created snapshot. Verify that the notification emails were received and contain information about the training. To further examine the results, download and view the model snapshot.

   Accessing and downloading the results of the training is easy.

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

5. **Create** a machine learning method from the newly created model snapshot.

   Creating methods from previously created snapshots is easy.

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

6. **Analyze** the provided patient cohort using the interface for the medical experts.

   The proposed abstraction of the analysis/inference process of beforehand generated machine learning models is clear and sufficient.

   ☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

Design & Workflow Evaluation:

Technical capability:

The here employed generic approach to modularize and apply pre- and postprocessing methods (virtualized application containers) could be used to implement the pre- and postprocessing methods used in my developed machine learning methods.

☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

The here employed generic approach to modularize and apply training methods (virtualized application containers) could be used to implement the training process used in my developed machine learning models.

☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

Overall impression:

very bad ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 very good

Guidance/Workflows:

hard-to-follow workflows ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 adequate guidance, clear instructions

Responsiveness:

lags, stuttering ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 no perceivable lags, fluid animations

Aesthetics:

poorly designed, unaesthetic ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 well designed, aesthetic

Purpose:
The proposed platform could ease the deployment of machine learning models and could, therefore, contribute to a closer collaboration between medical experts and clinical data scientists.

☐ strongly disagree ☐ disagree ☐ neutral ☐ agree ☐ strongly agree

Additional remarks:

Please add further comments, observations, or other feedback in the box below. If applicable, please add concrete examples to your comments. General thoughts and critics are very welcome: What is easy and why is it easy? What is difficult and why is it difficult?

Appendix B

Appendix: Detailed Evaluation Feedback

During evaluation each participant had an option to provide additional feedback. The additional feedback is provided in this chapter.

B.1 Additional Feedback from Medical Experts

In this section the additional feedback of the medical experts is shown.

Medical Expert 1:

- To further work with the results it would be better to have them stored in an imaging archive. The downloaded information is helpful for debugging reasons.
- Previously, it was quiet an effort to setup a segmentation method because many methods are very heterogenous. With such a platform it is much easier to test new methods and get results as fast as possible.

Medical Expert 2:

- Downloading: Allow download of segmentations only
- Very easy to apply since the the technical part is done automatically in the background
- Allows testing of many methods for a specific data set
- Analyze tab offers workflow as expected
- In methods section add action for *use method for analysis*

Medical Expert 3:

- In methods overview it would be beneficial to have an extra column for method/task category. So the developer is forced to specify it (e.g. segmentation, registration) instead of encoding it in the method name or description

- Viewing in platform is nice, but structure of archive could be improved / documentation should be provided.
- Basic statistics of results could be provided.

Medical Expert 4:

- From my experience, setting new standards within the scientific community is quite challenging.
- Nevertheless, I see great potential for such a ready-to-use platform as soon as enough people are using it.
- For me as beginner in machine learning I would be happy to use such a platform.

Medical Expert 5:

- Nice UI, easy to follow.
- Nice to have an introduction to machine learning as a new developer.
- Integration with slack and telegram.

Medical Expert 6:

- Place next button close to select tile button.
- Maybe viewing code is helpful.

Medical Expert 7:

- I find it easy to upload data and to apply specific methods on it.
- I think it would be useful to have a kind of notification before uploading to check possible requirements for the method.

Medical Expert 8:

- Include status bar in the results section.
- Disable debugging output in email for medical expert.
- Closes the gap for easy access to deep learning.

Medical Expert 9:

- Workflow is clear and easy to use.
- Result representation can be improved, less is more.
- With this platform machine learning application can be performed so efficiently: well organized, easy access, flexible to try different model in a short time.
- Good starting platform for a beginner to learn machine learning.

Medical Expert 10:

- Missing information about the method:
  - Modality, kind of data, resolution.
  - Kind of machine learning algorithm.
  - Trained on what kind and how much data.
  - Dice-Scores / accuracy on data set XY.
- Reduce debug output / make it optional.
- Licensing: what do I need to cite.
- Chat and a market place for methods.
- Upload user data prior to developer of model.

B.2 Additional Feedback from Clinical Data Scientist

In this section the additional feedback of the clinical data scientists is shown.

Clinical Data Scientist 1:

- Missing explanations of different Methods visible on the website (add tooltips).
- My model was easy to integrate during an alpha-test of the platform. This was additionally done aside from the walk-through included in this survey.

Clinical Data Scientist 2:

- Interface within the platform to view results could be an additional feature.
- When setting model parameters within the train function: selection could be more clear, setting check marks suggest that features are not enabled when unchecked, however it means they are enabled by default mode.
- Direct embedding of a computation cluster (shared server resources) could be helpful.

Clinical Data Scientist 3:

- Right now for a model its showing only the master branch. You could also show all other branches.
- The configuration of the model is fine but you can try to change (e.g. no of epochs) in a different way.
- You can also put some other available clusters option for the training.

Clinical Data Scientist 4:

- Expose debug options to frontend.
- Back button in available snapshots should go back to model version overview.
- GPU cluster integration.

Clinical Data Scientist 5:

- There should be a login form or email specific form.
- It would be good if we can see the status of the training at any time like number of epochs that are already done etc.
- platform should consider parameter optimization.
- Platform should have cluster capability, where one can use directly resources of the cluster.

Clinical Data Scientist 6:

- Connection to a cloud would be required to overcome limitations of a single workstation.
- Maybe would be nice for developers to get a tool for comparison of different methods.
- Very clear structure, easy to use without instructions

Clinical Data Scientist 7:

- In train: conformation of parameter settings with enter.
- In Snapshots: view/display hyperparameters used in training and add possibility to filter/search through trainings by hyperparameters.
- Possible usage of the cluster.

- Download: change structure to log folder with log-files for better overview.
- Applicable for my work with videos: pipeline enables easy use of complicated method for medical expert but leaves enough freedom to develop multistage process (preprocessing - converting to deep features application).
- Platform supports all necessary steps to set up and evaluate trainings.
- Required settings from developer on git facilitate usage of model (f.e. required packages needed for setting up environment on cluster or on local system of medical expert, hyperparameters for better overview).
- Further ideas: displaying log files or directly monitor if errors occur.

Clinical Data Scientist 8:

- Cluster integration.

Clinical Data Scientist 9:

- Going back from available snapshots to model version overview not working.
- Hide the IDs in training setup. Especially the non-human readable values.
- Keep all machine readable texts to one side in all the windows.
- Highlight the current step more (during training and application).
- Give a feedback once a method is created and also tell where to look.
- Message should be red or brighten so its immediately clear that it is a message to the user.
- Having an email/slack/social net notification is extremely useful for both medical experts and clinical data scientists.
- The platform makes the transfer and ease of technical communication for medical expert and clinical data scientist really streamlined (at least in theory).

Clinical Data Scientist 10:

- Great intuitive usability.
- Modern design.
- Git support is great.
- Cluster integrate could maybe be beneficial (with hyperparameter optimization).

Clinical Data Scientist 11:

- Would be good to add some documentation to the app itself. Just in case someone wants to check it.
- Also nice if the developer can expose his/her contact info in case users want to contact him/her to discuss model or even future work.

Clinical Data Scientist 12:

- Cluster would ease the usability.
- Increase size of download button.

Appendix C

Appendix: Further Details of Design and Implementation

The data structure is represented on storage in the following way:

C.1 Data Sets

C.1.1 Ideal Directory Structure

- Directory: **Dataset-UUID**
  - File: **dataset.json** This file contains all metadata and attributes of the dataset in the json format.
  - Directory: **training_data** This folder contains training data
  - Directory: **validation_data** This folder contains validation data
  - Directory: **test_data** This folder contains test data
  - Directory: **labels** This folder contains a label-file for each image of the training, validation, and test set.

C.1.2 Metadata File Attributes and Structure

- File: **dataset.json**
  - Data set identifier
  - Data set name
  - Data set description
  - Data set category (segmentation, detection, . . .)
  - Data set tags (project, group, . . .)
  - Standardized data set (true/false)
  - Maintainer
  - Version identifier
  - Custom metadata (like medical attributes)
  - List of training data (relative path)
  - List of validation data (relative path)
  - List of test data (relative path)

C.1.3 Main Functionality

- **List** (REQ-1): List all available data sets
- **Import** (REQ-2): Imports a data set to the platform.
- **Update** (REQ-3): Get updated data set from the same source as the input (only works if imported by URL), the data set is under version control.
- **Edit Metadata** (REQ-4): Edits the metadata of the data set.
- **Remove** (REQ-5): Removes a data set from the platform.

C.2 Machine Learning Models and Snapshots

- Directory: **Model-UUID**
  This directory contains the data corresponding to the model.
  - File: **model.json** This file contains all metadata and attributes of the model in the json format.
  - Directory: **source_code** This folder contains source code of the model. The source code is version controlled, deleting of a single version is not supported to preserve the consistency.
  - Directory: **snapshots** This folder contains the snapshots of the model.
    - Directory: **Snapshot-UUID**
      This folder contains the data corresponding to the snapshot.
      - File: **snapshot.json** This file specifies all attributes of the trained parameters in json format.
      - File: **hyperparameters.json** This file contains all hyperparameters.
      - File: **method.json** If this file exists, this snapshot is exposed and can be used as method for a specific tasks. It further defines the attributes of the exposed method. If the snapshot is deleted, the method must be deleted, too.
      - Directory: **Checkpoints**
        Best method weights achieved during training.
      - Directory: **Configuration**
        Used Method configuration for the training
      - Directory: **Sample Pictures**
        Sample pictures including annotations
      - Directory: **Training and Platform Logs**
        Log of each step of the training workflow.
      - Directory: **Plots**
        Loss function (training, validation and testing)
      - Directory: **Statistical Results**
        Detailed results and metrics for each training step.

C.2.1 Metadata File Attributes and Structure

- File: **model.json**
  - Model identifier
  - Model name
  - Model description
  - Model tags (Segmentation, Detection (location and classification), Registration, . . .)
  - Maintainer
  - Version identifier
  - Custom metadata (like medical attributes)
  - Hyperparameters
    Hyperparameters are modified by the user to adjust the training process, whereas parameters are usually learned during training.
  - Monitoring / Live tracking service settings
  - Default pre-processing pipeline
  - Default post-processing pipeline
- Directory: **Environment**
  - File: **environment.json**
    - Environment selection
    - Environment settings
  - File: **hardware-requirements.json**
    - Amount CPUs
    - Amount GPUs
    - Amount Memory
    - Amount Storage
    - CUDA Version
  - (optional) File: **Dockerfile**
  - (optional) File: **python-requirements.txt**
- File: **snapshot.json**
  - Snapshot identifier
    The snapshot identifier must be unique within the same version tag of the source code
  - Snapshot-Parent-UUID
    Identifier of the parent this snapshot was trained on. Unset if it was trained from scratch.
  - Model UUID
  - Model version UUID

  - Data set UUID
  - Data set version UUID
  - Container name
- File: **hyperparameters.json**
  - Snapshot identifier
  - Definition of all hyperparameters used for this snapshot.

C.2.2 Main Functionality

- **List** (REQ-6): Lists all available models.
- **Import** (REQ-7): Imports a model source code repository to the platform.
- **Update** (REQ-8): Get updated model from the same source as the input the models source code is under version control.
- **Edit Metadata** (REQ-9): Edits the metadata of the model.
- **Remove** (REQ-10): Removes a model from the platform.
- **List Snapshots** (REQ-11): List all snapshots.
- **Remove Snapshots** (REQ-12): Remove a snapshot.
- **Train Model** (REQ-24): Creates a new model snapshot. This also includes the tasks *Subscribe to updates* (REQ-17), *Monitor and terminate* (REQ-19), *View and download results* (REQ-20), *Track progress* (REQ-22), *Customize preprocessing pipeline* (REQ-23), *Initialize training using snapshot* (REQ-24), and *Customize hyperparameters* (REQ-25).

C.3 Machine Learning Methods and Results

C.3.1 Ideal Method Directory Structure

The metadata of the method is stored in the *method.json* file, in the same directory as the used snapshot of the model.

After each application a new result object is created which stores the following information:

- Patient data (copy of uploaded data and available metadata)
- Generated output from the method (annotated data)
- Logs from the processing pipeline

C.3.2 Metadata File Attributes and Structure

- File: **method.json**
  - Method identifier
  - Method name
  - Method description
  - Method tags (Segmentation, Detection (location and classification), Registration, . . .)
  - Maintainer
  - Custom metadata (like medical attributes)
  - Model identifier
  - Parameter set identifier
  - Default pre-processing pipeline (e.g. DICOM -> Methods format)
  - Default post-processing pipeline (e.g. Methods format -> DICOM)
  - Success/Fail Status, including error message in case of failure

C.3.3 Main Functionality

Main Functionality for creating and deleting methods is handled by the model functionality. The method itself should not need any configuration except the connection to the image data.

- **List Methods** (REQ-13): Show all available methods.
- **Create** (REQ-14): Create a new method based on a model snapshot.
- **Remove Method** (REQ-15): Remove a method.
- **Import Patient Data** (REQ-16): Import patient data for automated inference.
- **Automated Inference** (REQ-18): Run inference on a given patient data. This also includes the tasks *Subscribe to updates* (REQ-17), *Monitor and terminate* (REQ-19), and *View and download results* (REQ-20)

Bibliography

[1] Martín Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning". In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Savannah, GA, USA). OSDI'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 265–283. URL: `http://dl.acm.org/citation.cfm?id=3026877.3026899` (visited on 02/13/2019).

[2] S. Albarqouni et al. "AggNet: Deep Learning From Crowds for Mitosis Detection in Breast Cancer Histology Images". In: *IEEE Transactions on Medical Imaging* 35.5 (May 2016), pp. 1313–1321.

[3] *Amazon ECS - Run Containerized Applications in Production*. URL: `https://aws.amazon.com/ecs/` (visited on 08/09/2019).

[4] *Amazon EKS - Managed Kubernetes Service*. URL: `https://aws.amazon.com/eks/` (visited on 06/07/2019).

[5] *An Open Source Machine Learning Framework for Everyone: Tensorflow/Tensorflow*. Feb. 13, 2019. URL: `https://github.com/tensorflow/tensorflow` (visited on 02/13/2019).

[6] André Anjos, Laurent El-Shafey, and Sébastien Marcel. "BEAT: An Open-Source Web-Based Open-Science Platform". In: *arXiv:1704.02319 [cs]* (Apr. 7, 2017). URL: `http://arxiv.org/abs/1704.02319` (visited on 02/05/2019).

[7] National Electrical Manufacturers Association. "Digital Imaging and Communications in Medicine (DICOM)". In: *http.//medical.nema.org/* (2003). URL: `https://ci.nii.ac.jp/naid/10028228113/` (visited on 02/13/2019).

[8] Deepika Badampudi, Claes Wohlin, and Kai Petersen. "Experiences from Using Snowballing and Database Searches in Systematic Literature Studies". In: *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering - EASE '15*. The 19th International Conference. Nanjing, China: ACM Press, 2015, pp. 1–10. URL: `http://dl.acm.org/citation.cfm?doid=2745802.2745818` (visited on 01/28/2019).

[9] Robert Battle and Edward Benson. "Bridging the Semantic Web and Web 2.0 with Representational State Transfer (REST)". In: *Journal of Web Semantics*. Semantic Web and Web 2.0 6.1 (Feb. 1, 2008), pp. 61–69. URL: `http://www.sciencedirect.com/science/article/pii/S1570826807000510` (visited on 06/07/2019).

[10] Kent Beck. *Test-Driven Development: By Example*. Addison-Wesley Professional, 2003. 241 pp.

[11] Andrew Beers et al. "DeepNeuro: An Open-Source Deep Learning Toolbox for Neuroimaging". In: *arXiv:1808.04589 [cs]* (Aug. 14, 2018). URL: `http://arxiv.org/abs/1808.04589` (visited on 02/06/2019).

[12] Erwin Bellon et al. "Trends in PACS Architecture". In: *European Journal of Radiology*. From PACS to the Clouds 78.2 (May 1, 2011), pp. 199–204. URL: http://www.sciencedirect.com/science/article/pii/S0720048X10002408 (visited on 02/13/2019).

[13] D. Bernstein. "Containers and Cloud: From LXC to Docker to Kubernetes". In: *IEEE Cloud Computing* 1.3 (Sept. 2014), pp. 81–84.

[14] Anant Bhardwaj et al. "DataHub: Collaborative Data Science & Dataset Version Management at Scale". In: *arXiv:1409.0798 [cs]* (Sept. 2, 2014). URL: http://arxiv.org/abs/1409.0798 (visited on 01/24/2019).

[15] Souvik Bhattacherjee et al. "Principles of Dataset Versioning: Exploring the Recreation/Storage Tradeoff". In: *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases* 8.12 (Aug. 2015), pp. 1346–1357. pmid: 28752014.

[16] Carl Boettiger. "An Introduction to Docker for Reproducible Research". In: *SIGOPS Oper. Syst. Rev.* 49.1 (Jan. 2015), pp. 71–79. URL: http://doi.acm.org/10.1145/2723872.2723882 (visited on 02/19/2019).

[17] *Brain Team*. URL: https://ai.google/research/teams/brain/ (visited on 02/13/2019).

[18] Eric A. Brewer. "Kubernetes and the Path to Cloud Native". In: *Proceedings of the Sixth ACM Symposium on Cloud Computing*. SoCC '15. New York, NY, USA: ACM, 2015, pp. 167–167. URL: http://doi.acm.org/10.1145/2806777.2809955 (visited on 01/18/2019).

[19] David Budgen and Pearl Brereton. "Performing Systematic Literature Reviews in Software Engineering". In: *Proceeding of the 28th International Conference on Software Engineering - ICSE '06*. Proceeding of the 28th International Conference. Shanghai, China: ACM Press, 2006, p. 1051. URL: http://portal.acm.org/citation.cfm?doid=1134285.1134500 (visited on 01/28/2019).

[20] Peter Buneman et al. "Archiving Scientific Data". In: *ACM Trans. Database Syst.* 29.1 (Mar. 2004), pp. 2–42. URL: http://doi.acm.org/10.1145/974750.974752 (visited on 02/05/2019).

[21] Joaquin Quiñonero Candela. *Facebook and Microsoft Introduce New Open Ecosystem for Interchangeable AI Frameworks*. URL: https://research.fb.com/facebook-and-microsoft-introduce-new-open-ecosystem-for-interchangeable-ai-frameworks (visited on 02/13/2019).

[22] Ryan Chard et al. "DLHub: Model and Data Serving for Science". In: *arXiv:1811.11213 [cs, stat]* (Nov. 27, 2018). URL: http://arxiv.org/abs/1811.11213 (visited on 02/06/2019).

[23] Lawrence Chung et al. *Non-Functional Requirements in Software Engineering*. Springer Science & Business Media, Dec. 6, 2012. 458 pp.

[24] Alan Cooper. *The Inmates Are Running the Asylum*. Indianapolis, IN: Sams, 2004. 255 pp.

[25] NVIDIA Corporation. *Build and Run Docker Containers Leveraging NVIDIA GPUs: NVIDIA/Nvidia-Docker*. July 21, 2019. URL: `https://github.com/NVIDIA/nvidia-docker` (visited on 07/21/2019).

[26] Daniel Crankshaw et al. "The Missing Piece in Complex Analytics: Low Latency, Scalable Model Management and Serving with Velox". In: (), p. 7.

[27] Organización Internacional de Normalización. *ISO-IEC 25010: 2011 Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-System and Software Quality Models*. ISO, 2011.

[28] Anh Dinh et al. "UStore: A Distributed Storage With Rich Semantics". In: *arXiv:1702.02799 [cs]* (Feb. 9, 2017). URL: `http://arxiv.org/abs/1702.02799` (visited on 02/06/2019).

[29] Tom Doel et al. "GIFT-Cloud: A Data Sharing and Collaboration Platform for Medical Imaging Research". In: *Computer Methods and Programs in Biomedicine* 139 (Feb. 1, 2017), pp. 181–190. URL: `http://www.sciencedirect.com/science/article/pii/S016926071630654X` (visited on 02/05/2019).

[30] EastBancTech. *Building a Reliable Kubernetes Cluster in the Amazon Cloud*. July 16, 2017. URL: `https://blog.kublr.com/building-a-reliable-kubernetes-cluster-in-the-amazon-cloud-5543dffb297d` (visited on 07/16/2017).

[31] I. Eggel, R. Schaer, and H. Müller. "Distributed Container-Based Evaluation Platform for Private/Large Datasets". In: *2018 17th International Symposium on Parallel and Distributed Computing (ISPDC)*. 2018 17th International Symposium on Parallel and Distributed Computing (ISPDC). June 2018, pp. 93–100.

[32] W. Felter et al. "An Updated Performance Comparison of Virtual Machines and Linux Containers". In: *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Mar. 2015, pp. 171–172.

[33] Jesse James Garrett. *Elements of User Experience,The: User-Centered Design for the Web and Beyond*. Pearson Education, Dec. 16, 2010. 226 pp.

[34] Eli Gibson et al. "NiftyNet: A Deep-Learning Platform for Medical Imaging". In: *Computer Methods and Programs in Biomedicine* 158 (May 2018), pp. 113–122. pmid: 29544777.

[35] Kurt Griffiths et al. "Falcon Documentation". In: (2015).

[36] Alon Halevy et al. "Goods: Organizing Google's Datasets". In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. New York, NY, USA: ACM, 2016, pp. 795–806. URL: `http://doi.acm.org/10.1145/2882903.2903730` (visited on 02/06/2019).

[37] Allan Hanbury, Henning Müller, and Georg Langs. *Cloud-Based Benchmarking of Medical Image Analysis*. OCLC: 988540448. 2017. URL: `http://dx.doi.org/10.1007/978-3-319-49644-3` (visited on 02/05/2019).

[38] Allan Hanbury et al. “Bringing the Algorithms to the Data: Cloud–Based Benchmarking for Medical Image Analysis”. In: *Information Access Evaluation. Multilinguality, Multimodality, and Visual Analytics*. Ed. by Tiziana Catarci et al. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 24–29.

[39] Allan Hanbury et al. “Cloud–Based Evaluation Framework for Big Data”. In: *The Future Internet*. Ed. by Alex Galis and Anastasius Gavras. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 104–114.

[40] Allan Hanbury et al. “Evaluation-as-a-Service: Overview and Outlook”. In: *arXiv:1512.07454 [cs]* (Dec. 23, 2015). URL: `http://arxiv.org/abs/1512.07454` (visited on 02/05/2019).

[41] K. Hazelwood et al. “Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective”. In: *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). Feb. 2018, pp. 620–629.

[42] Benjamin Hindman et al. “Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center.” In: *NSDI*. Vol. 11. 2011, pp. 22–22.

[43] Computer Society IEEE. “Data Engineering: Special Issue on Machine Learning Life-Cycle Management”. In: Vol. 41 No. 4 (Dec. 2018), p. 64.

[44] Samireh Jalali and Claes Wohlin. “Systematic Literature Studies: Database Searches vs. Backward Snowballing”. In: *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '12*. The ACM-IEEE International Symposium. Lund, Sweden: ACM Press, 2012, p. 29. URL: `http://dl.acm.org/citation.cfm?doid=2372251.2372257` (visited on 01/28/2019).

[45] *JupyterLab Documentation — JupyterLab 1.0.3 Documentation*. URL: `https://jupyterlab.readthedocs.io/en/stable/` (visited on 07/26/2019).

[46] Dharmesh Kakadia. *Apache Mesos Essentials*. Packt Publishing Ltd, June 29, 2015. 230 pp.

[47] Brian Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice hall, 2017.

[48] Nikhil Ketkar. “Introduction to PyTorch”. In: *Deep Learning with Python: A Hands-on Introduction*. Ed. by Nikhil Ketkar. Berkeley, CA: Apress, 2017, pp. 195–208. URL: `https://doi.org/10.1007/978-1-4842-2766-4_12` (visited on 06/11/2019).

[49] B. Kitchenham and S. Charters. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. 2007.

[50] Barbara Kitchenham et al. “Systematic Literature Reviews in Software Engineering – A Systematic Literature Review”. In: *Information and Software Technology* 51.1 (Jan. 2009), pp. 7–15. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0950584908001390` (visited on 01/28/2019).

[51] Gerald Kotonya and Ian Sommerville. *Requirements Engineering: Processes and Techniques*. 1st. Wiley Publishing, 1998.

[52] A. Kovari and P. Dukan. “KVM Amp; OpenVZ Virtualization Based IaaS Open Source Cloud Virtualization Platforms: OpenNode, Proxmox VE”. In: *2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics*. 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics. Sept. 2012, pp. 335–339.

[53] Konstantinos Krampis et al. “Cloud BioLinux: Pre-Configured and on-Demand Bioinformatics Computing for the Genomics Community”. In: *BMC Bioinformatics* 13.1 (Mar. 19, 2012), p. 42. URL: `https://doi.org/10.1186/1471-2105-13-42` (visited on 07/20/2019).

[54] S. P. T. Krishnan and Jose L. Ugia Gonzalez. “Google Compute Engine”. In: *Building Your Next Big Thing with Google Cloud Platform: A Guide for Developers and Enterprise Architects*. Ed. by S. P. T. Krishnan and Jose L. Ugia Gonzalez. Berkeley, CA: Apress, 2015, pp. 53–81. URL: `https://doi.org/10.1007/978-1-4842-1004-8_4` (visited on 02/18/2019).

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf` (visited on 08/15/2019).

[56] T. Küstner et al. “An Easy-to-Use Image Labeling Platform for Automatic Magnetic Resonance Image Quality Assessment”. In: *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017). Apr. 2017, pp. 754–757.

[57] Alex Kyriakidis and Kostas Maniatis. *The Majesty of Vue.Js*. Packt Publishing Ltd, Nov. 14, 2016. 230 pp.

[58] S.- B. Lo et al. “Artificial Convolution Neural Network Techniques and Applications for Lung Nodule Detection”. In: *IEEE Transactions on Medical Imaging* 14.4 (Dec. 1995), pp. 711–718.

[59] Jochen Ludewig and Horst Lichter. *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken*. dpunkt.verlag, May 17, 2013. 846 pp.

[60] Sanket Meghani. *JavaScript Linting Tools Comparison*. June 14, 2019. URL: `https://codeburst.io/javascript-linting-tools-comparison-ebcb4aa23c49` (visited on 06/14/2019).

[61] Daniel A. Menascé. “Virtualization: Concepts, Applications, and Performance Modeling”. In: *Int. CMG Conference*. 2005.

[62] H. Miao et al. “ModelHub: Deep Learning Lifecycle Management”. In: *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 2017 IEEE 33rd International Conference on Data Engineering (ICDE). Apr. 2017, pp. 1393–1394.

[63] H. Miao et al. “Towards Unified Data and Lifecycle Management for Deep Learning”. In: *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 2017 IEEE 33rd International Conference on Data Engineering (ICDE). Apr. 2017, pp. 571–582.

[64] F. Milletari et al. “Cloud Deployment of High-Resolution Medical Image Analysis with TOMAAT”. In: *IEEE Journal of Biomedical and Health Informatics* (2018), pp. 1–1.

[65] mjenkinson. *NIfTI-1 Data Format — Neuroimaging Informatics Technology Initiative*. URL: `https://nifti.nimh.nih.gov/nifti-1/` (visited on 02/14/2019).

[66] Rimantas Mocevicius. *CoreOS Essentials*. Packt Publishing Ltd, June 29, 2015. 132 pp.

[67] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. 2019.

[68] Kief Morris. *Infrastructure as Code: Managing Servers in the Cloud*. "O’Reilly Media, Inc.", June 9, 2016. 362 pp.

[69] Alison Nightingale. “A Guide to Systematic Literature Reviews”. In: *Surgery (Oxford)* 27.9 (Sept. 2009), pp. 381–384. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0263931909001707` (visited on 01/28/2019).

[70] Bashar Nuseibeh and Steve Easterbrook. “Requirements Engineering: A Roadmap”. In: *Proceedings of the Conference on The Future of Software Engineering - ICSE ’00*. The Conference. Limerick, Ireland: ACM Press, 2000, pp. 35–46. URL: `http://portal.acm.org/citation.cfm?doid=336512.336523` (visited on 02/19/2019).

[71] T. E. Oliphant. “Python for Scientific Computing”. In: *Computing in Science Engineering* 9.3 (May 2007), pp. 10–20.

[72] Barbara Paech and Kirstin Kohler. “Task-Driven Requirements in Object-Oriented Development”. In: *Perspectives on Software Requirements*. Ed. by Julio Cesar Sampaio do Prado Leite and Jorge Horacio Doorn. The Springer International Series in Engineering and Computer Science. Boston, MA: Springer US, 2004, pp. 45–67.

[73] *PEP 8 – Style Guide for Python Code*. URL: `https://www.python.org/dev/peps/pep-0008/` (visited on 06/14/2019).

[74] *Postman | API Development Environment*. URL: `https://www.getpostman.com` (visited on 07/16/2019).

[75] *PyTorch*. URL: `https://www.pytorch.org` (visited on 02/14/2019).

[76] Suzanne Robertson and James Robertson. *Mastering the Requirements Process: Getting Requirements Right*. Addison-Wesley, Aug. 6, 2012. 579 pp.

[77] Rami Rosen. “Linux Containers and the Future Cloud”. In: *Linux J.* 2014.240 (Apr. 2014). URL: `http://dl.acm.org/citation.cfm?id=2618216.2618219` (visited on 02/18/2019).

[78] Chris Rupp and Klaus Pohl. *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant*. 2nd ed. Santa Barbara, CA: Rocky Nook, May 5, 2015. 164 pp.

[79] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. "OpenStack: Toward an Open-Source Solution for Cloud Computing". In: *International Journal of Computer Applications* 55.3 (Oct. 20, 2012), pp. 38–42. URL: `http://research.ijcaonline.org/volume55/number3/pxc3882991.pdf` (visited on 03/27/2019).

[80] E. Serrano et al. "Medical Imaging Processing on a Big Data Platform Using Python: Experiences with Heterogeneous and Homogeneous Architectures". In: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). May 2017, pp. 830–837.

[81] Keith Seymour et al. "Overview of GridRPC: A Remote Procedure Call API for Grid Computing". In: *International Workshop on Grid Computing*. Springer. 2002, pp. 274–278.

[82] Prateek Sharma et al. "Containers and Virtual Machines at Scale: A Comparative Study". In: *Proceedings of the 17th International Middleware Conference* (Trento, Italy). Middleware '16. New York, NY, USA: ACM, 2016, 1:1–1:13. URL: `http://doi.acm.org/10.1145/2988336.2988337` (visited on 02/18/2019).

[83] *Teem: Nrrd*. URL: `http://teem.sourceforge.net/nrrd/index.html` (visited on 02/14/2019).

[84] S. Tilkov and S. Vinoski. "Node.Js: Using JavaScript to Build High-Performance Network Programs". In: *IEEE Internet Computing* 14.6 (Nov. 2010), pp. 80–83.

[85] A. Tosatto, P. Ruiu, and A. Attanasio. "Container-Based Orchestration in Cloud: State of the Art and Challenges". In: *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*. 2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems. July 2015, pp. 70–75.

[86] Muhammad Umair. *SOLID, GRASP, and Other Basic Principles of Object-Oriented Design*. URL: `https://dzone.com/articles/solid-grasp-and-other-basic-principles-of-object-o` (visited on 08/12/2019).

[87] *Using Radon Programmatically — Radon 2.4.0 Documentation*. URL: `https://radon.readthedocs.io/en/latest/api.html#module-radon.raw` (visited on 08/01/2019).

[88] Manasi Vartak et al. "ModelDB: A System for Machine Learning Model Management". In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. HILDA '16. New York, NY, USA: ACM, 2016, 14:1–14:3. URL: `http://doi.acm.org/10.1145/2939502.2939516` (visited on 01/31/2019).

[89] Viswanath Venkatesh and Fred D Davis. "A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies". In: *Management science* 46.2 (2000), pp. 186–204.

[90] *Vue Native*. URL: `https://vue-native.io/` (visited on 03/28/2019).

[91] *Vue.Js Material Component Framework — Vuetify.Js*. URL: `https://vuetifyjs.com/en/` (visited on 06/11/2019).

[92] *Welcome to Radon's Documentation! — Radon 2.4.0 Documentation*. URL: `https://radon.readthedocs.io/en/latest/index.html` (visited on 08/01/2019).

[93] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer, Nov. 19, 2014. 327 pp.

[94] Bill Wilder. *Cloud Architecture Patterns: Using Microsoft Azure*. "O'Reilly Media, Inc.", Sept. 20, 2012. 183 pp.

[95] Alex Williams. *CI/CD with Kubernetes*. The New Stack, Dec. 11, 2018.

[96] Andreas Wittig and Michael Wittig. *Amazon Web Services in Action*. 1st. Greenwich, CT, USA: Manning Publications Co., 2015.

[97] Claes Wohlin. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering". In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*. The 18th International Conference. London, England, United Kingdom: ACM Press, 2014, pp. 1–10. URL: `http://dl.acm.org/citation.cfm?doid=2601248.2601268` (visited on 01/28/2019).

[98] Claes Wohlin et al. *Experimentation in Software Engineering*. Springer Science & Business Media, June 16, 2012. 249 pp.

[99] M. G. Xavier et al. "Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments". In: *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. Feb. 2013, pp. 233–240.

[100] Y. Zhang et al. "DataLab: A Version Data Management and Analytics System". In: *2016 IEEE/ACM 2nd International Workshop on Big Data Software Engineering (BIGDSE)*. 2016 IEEE/ACM 2nd International Workshop on Big Data Software Engineering (BIGDSE). May 2016, pp. 12–18.