Monitoring with MLOps for Clinical Decision Support

Adam Ekblom

LUND UNIVERSITY

GETINGE

BMEM01
Master's Thesis in Biomedical Engineering
2024
Academic Supervisor: Martin Stridh
Industrial Supervisors: Jonas Andersson L., Lena Mondrejevski
LTH, Faculty of Engineering, Department of Biomedical Engineering
Sweden

Abstract

The development of Artificial Intelligence(AI) and Machine Learning(ML) has highly increased over the past decades, with numerous research projects developing AI and ML models for clinical decision support. Many of these projects showcase promising results in the lab, however, when set in a real-life clinical setting most models perform significantly worse, resulting in only a few models transitioning from a lab environment to clinical use. One reason of many for the discrepancy between research and commercial application is the absence, and thereof need, for established development processes that deploy and monitor ML models for clinical use in hospital environments.
This thesis aimed to implement a model monitoring framework utilizing Machine Learning Operations(MLOps), a development process focusing on streamlining development, monitoring, and maintenance, to detect performance variations in clinical decision support systems. The monitoring framework was implemented and evaluated by creating a use-case utilizing the MIMIC-IV-2.2 data set, which comprises real-life patient data from electronic health records, for training, validating, and deploying a novel neural network model that predicts the Length of Stay(LoS) of a patient admitted to the Intensive Care Unit (ICU) of a hospital. The data was split into training- and inference data, based on age groups, where the former was used for model training, and the latter for evaluating real-life performance via model monitoring. Finally, the performance of the model was evaluated by a set of metrics to determine if performance drift occurred with the inference data. Resultingly, an end-to-end framework was implemented with successful monitoring, able to detect a significant performance drift and loss of performance. The more significant metrics, Mean Square Log Error (MSLE) and Mean Absolute Percentage Error (MAPE) detected a performance loss of 44.64% and 41.13% respectively, compared to model performance on the training data. The MLOps framework used in the thesis fulfilled the use-case's intentions, had a clear structure, and showed efficiency. However, there are limitations involving no retraining of the ML model implemented in the use-case, the disparity from a real-life clinical environment, and whether the framework used for the thesis is appropriate for similar development processes, illustrating questions that need to be addressed with further research. The MLOps framework however shows promise, employing tools and practices that can further advance the transition, and integration of AI and ML-based technology in clinical environments.

Preface

This master's thesis was conducted at Getinge Critical Care throughout 2023. The opportunity to collaborate with the intelligent people at the Solna office has been rewarding and challenging, with people eager to assist if needed. I want to express in particular sincere gratitude to Jonas Andersson L. and Lena Mondrejevski, my supervisors for the project. Your insightful and helpful comments stemming from your vast knowledge in the field have been paramount to the process and helped to shape the thesis to where it is today. I would also like to express gratitude to Martin Stridh, my supervisor at the Department of Biomedical Engineering at LTH, Faculty of Engineering.

Contents

List of Figures

List of Tables

1

Introduction

Clinical Decision Support Systems (CDSS) is a technical software system used to assist clinicians in complex diagnostic decisions for an optimal patient outcome, based on diagnostic measurements in a clinical setting. Since the first use of the technology in the 1970s, current CDSSs are used in web applications, with electronic health records, computerized provider order entry systems, and administration in almost all interactive healthcare technology today. For the longer duration of its usage, the CDSS has used a rule-based system, programmed IF-THEN-statements that are based on a set of rules or values. These rules can be knowledge-based, such as literature-based, value-based, or practice-based, and programmed to follow expert medical knowledge[Sut+20]. Over the last decade, more non-knowledge-based CDSS have been appearing, based on Artificial Intelligence(AI) and Machine Learning(ML) algorithms, that utilize clinical data rather than programmed medical knowledge. As more historical health data sets become available, the usage of ML algorithms follows suit, being more adopted and integrated[Res21]. These data enable the use of deep learning techniques in CDSS and the rise of predictive analytics, methods that use advanced statistics and ML techniques to predict future events[McC+22].

However, the process from an idea to a potential ML application to a fully developed product is a very complicated task. Up to 85% of all AI- or ML projects fail to reach a finalized product[MM18]. For clinical ML applications, in this case for COVID-19 diagnosis, the statistics are worse[Wyn+20]. The major reason for failure is mainly credited to unclear objectives for the end application and faulty research and development processes.

The latter reason can be attributed to that ML research and development have been primarily driven by teams consisting of data scientists, and not by software engineer teams. This has resulted in major leaps in development in the ML field but has resulted in a discrepancy from the standards of today's software development processes[ME22]. As ML applications become a more integrated part of our human-decision making, larger demands are set on the real-world performance of the models, and establishing standard processes for the projects that develop them, particularly in the field of healthcare, where the decision-making potentially directly affects a patient's health and future well-being. Regulatory measures for AI applications are already being implemented. Therefore established processes and practices for development and

operations management that handle the multi-disciplinary challenges of ML projects are now needed more than ever. There are ML applications for healthcare that do reach commercial usage, specifically in the US. However, due to old regulations, the ML models that reach clinical platforms are locked from retraining[FA21a]. However, one of 10 new guiding principles for ML development in healthcare, written by the Food and Drug Administration (FDA), mentions the need for deployed models to be monitored and to manage retraining risks for improved performance and safety[FA21b]. This opens new possibilities for model monitoring, and exploring the use of novel practices for ML deployment in clinical decision support.

1.1 Purpose of the thesis

The project aims to implement a model monitoring and maintenance framework by utilizing resources from a Machine learning Operations(MLOps) framework, or pipeline, to monitor the performance and detect performance drift of ML models for CDSS. MLOps is a development process focused on streamlining the deployment of ML models to a live service, where model monitoring is a major theme. The expected contribution of the project is furthering the development of digital autonomous integration of Getinge's products, contributing to further digitalization in healthcare, and advancement of knowledge in ML for CDSS.

1.2 Limitations

The project's main goal is to detect performance variation with a monitoring- and maintenance framework for ML models in CDSS. The initial quality of the model after training is not an objective of the thesis. The focus is the development and change of the model predictions for their suited task when exposed to altered clinical data. This project focuses on model monitoring with a certain set of programs and monitoring parameters applicable for CDSS, in this use-case, the ICU. More or less success can be achieved with other resources or other ML model tasks for the ICU or other wards. The project does not offer a final application for a fully implementable MLOps framework, but rather a proof-of-concept for monitoring of CDSS and an evaluation of the development framework.

1.3 Outline of the Thesis

Firstly some background knowledge is presented in Section 2, where the theory used for the proposed case study is presented, and to familiarise the reader with Machine Learning and its usage in healthcare applications. The methodology is presented in Section 3, with its appurtenant subtasks. The results are presented in Section 4 and are presented, for certain parts of the method subtasks, in a graphical manner such as plots and images. A discussion is presented in Section 5, where the used methodology and the results will be further evaluated. Finally, conclusions from the thesis are presented in Section 6. Ethical aspects of the project are discussed in Section 7.

2

Background

2.1 Machine Learning

Machine Learning (ML) is part of the Artificial Intelligence(AI) field. The term was introduced by the American researcher Arthur Samuel in 1959 as a tool to empower computers to learn from explicit data. The term was further elaborated by Tom Mitchell, in 1997, as a method in which the machine performs a task, measures, and evaluates the performance over task iterations to gain experience and optimize the performance[Sin22]. The essence of machine learning is, based on data with an unknown probability distribution, to construct a model from the data occurrences that can accurately predict the outcome of new data occurrences. The ability to accurately predict outcomes is essential to ML, where the model's architecture has to be suitable to the new data. The data points are often stored in vectors of "features". A feature is a property or a parameter of a data set. These features are used as input variables for the model to train, validate, and test its performance. A high-quality feature vector is highly important to achieve good results from the machine learning process, therefore much emphasis is placed on feature engineering over the machine learning workflow. Examples of feature engineering methods are normalization of the data, dimension reduction, and encoding [Hea17].

Target- or output variables from the data set are named label data. This data is associated with a higher-level fact or quantity of interest, such as the expected price of housing, or determining the existence of a tumor in mammography screenings. A label can be explained as a property of a data point that can only be determined with some type of human expertise, and the goal of most Machine Learning models is to predict the correct label based on the available features. Labels can be numerical for regression problems, categorical for classification problems, or ordinal for a combination of both former examples [Jun22].

The machine learning method, or the model, has the informal principle to construct from features $x \rightarrow X$ and labels $y \rightarrow Y$ a hypothesis space such that the hypothesis generated from one or several characteristic values is approximately equal to the label value, i.e. $h : X \rightarrow Y$ such that:

$$h(x) \approx \hat{y}$$

The machine learning method used is based on the nature of the input and

output data. There are four different types of machine learning methods, that are considered broad categories, *supervised learning*, *semi-supervised learning*, *unsupervised learning*, and *reinforcement learning*. In supervised learning, there are input features and corresponding output labels, and the goal of the model is to determine a general hypothesis that maps the correct features to the proper labels. Unsupervised learning means the data are unlabelled and the model has to construct its own hypothesis. This can be used to find hidden patterns in the data. In semi-supervised learning, there are input features, but a smaller number of output labels. it combines supervised- and unsupervised learning techniques that handle both labeled and unlabeled data. Reinforcement learning is an approach where the machine learning method interacts with a dynamic environment, and while acting in it, will provide feedback to the model in a reward-based manner, based on the model's actions. The model will strive to achieve the maximum reward while navigating this space[Mir22].

2.1.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) or *Neural Nets* is a vast sub-part of the machine learning field. It is a computational system that mimics the function of the biological neural networks that constitute the animal brain. The first computational neural network, based on the neural principles defined by Donald Hebb [Sha86], was modeled by W. Clark and B. Farley, documenting the earliest machine learning task with a neural network[FC54].
A neural network can be characterized as a graph with a set of nodes connected with edges. A visual illustration of a fundamental Neural network structure is seen in Fig. 2.1. The initial nodes are the input values transmitted through the network. The edges apply weights, a multiplicative factor, and biases, an additive factor, to the input value. This is defined as a linear transformation of the input. The nodes further apply an activation function, a linear or non-linear function that determines whether the node should be activated or not based on the sum of inputs and added bias. Depending on the function, The output is usually in the interval of 0 and 1, or -1 and 1. See equation (2.1) for a typical activation function, the sigmoidal.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

These operations follow in a sequence of transformations through several node layers that result in an output function, that can be either a classification decision or a value prediction, as seen in equation (2.2).

$$Y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.2}$$

The network is trained and modified by altering the weights and biases connected to the nodes after each pass forward in the network. The alteration is based on back-propagation. Back-propagation is the calculation of the gradient for the loss function, the difference between predicted values and label values, for each respective weight. Loss functions look different based on what type of task the network has, such as Mean Squared Error (2.3) for regression

tasks, or Cross-Entropy-Loss (2.4) for classification tasks. The loss function tells us how much the parameters need to be adjusted for further minimization of the functions, i.e. decreasing the difference between predicted- and real output.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.3}$$

$$CE = -\sum_{i=1}^{n}[Y_i \log(\hat{Y}_i) + (1 - Y_i)\log(1 - \hat{Y}_i)] \tag{2.4}$$

With the gradient, the network can be optimized by tuning the weights properly and informing where the most optimized weights for the best performance are found. The optimization of the weights is done with an optimization algorithm. The optimization of the weights is where the training of the ANN takes place. See equation (2.5) for a common optimization function, the gradient descent. θ_t is the parameter vector, α is the learning rate constant, $\nabla J(\theta_t)$ is the gradient of the cost function J, concerning θ_t.

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t) \tag{2.5}$$

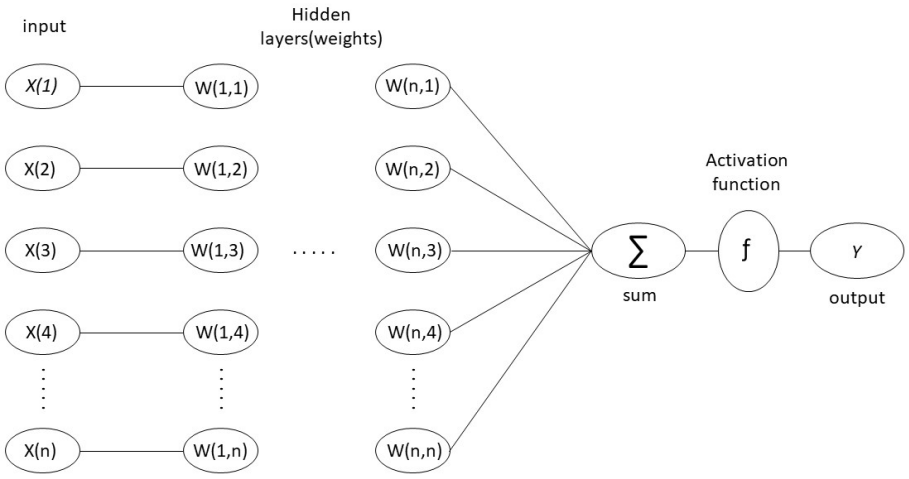

Figure 2.1: Graph structure of a multi-layer perceptron, a fundamental neural network structure

2.1.2 Convolutional Neural Network

Convolutional Neural Networks(CNNs), or Convolutional Networks, is a neural network specializing in reading and performing operations on input data with a grid-like topology, like matrices with arbitrary dimensions, two-dimensional images, or one-dimensional time-series data. The first Convolutional Network architecture was presented by Lecun et al. in 1989. It used novel techniques of the time, such as incorporating weight constraint, geometric knowledge of the input in the task domain, and applied back-propagation. The model architecture saw fast commercial success in reading handwritten postal zip codes [LeC+89]. The name convolutional neural networks stem from applying the convolution operation inside the network. To put it simply, A convolutional network is a network that uses convolution once or more in its layers. See equation (2.6).

The convolution operation is a linear operation between two functions with real-valued arguments that produces a third function based on the functions' shapes.

In a CNN the two components of the operation are the input matrix, which can be, for example, the initial image data or the output of another layer, and the convolutional filter, or kernel, which is a smaller matrix, with a value distribution to extract certain matrix features. The result of the operation creates a feature map.

$$(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[n-k] \tag{2.6}$$

The convolution operation introduces three concepts for a more efficient network, **sparse interactions**, **parameter sharing**, and **equivariant connectivity**. Sparse interactions make the kernel in the convolution operation smaller in dimension than the input. In a typical neural network layer, matrix multiplication is used, with a matrix filled with separate parameters describing each input unit's relation to the output unit. With n inputs and m outputs, a matrix multiplication of $n \times m$ would have a runtime of $O(n \times m)$. However, if introducing a kernel of smaller size k, the resulting convolution operation has a shorter runtime of $O(n \times k)$. So for example when processing a very large image, with millions of pixels as input, we can detect meaningful features such as edges with kernels that only consist of tens or hundreds of pixels. This results in the ability to describe complex interactions while obtaining good performance. Parameter sharing means that the value of a weight applied to an input is also applied elsewhere, causing only one set of parameters to be learned for a location, affecting the storage requirements of the model. As the number of stored parameters is reduced by a large margin, compared to matrix multiplication, the performance in memory requirements and statistical efficiency is improved. Parameter sharing also makes the layer in the network have equivariant connectivity, meaning that, for example, a function *f(x)* is equivariant to the function *g(x)* if $f(g(x)) = g(f(x))$. This means that if the input would change, the output will change identically. The convolution operation is followed by a set of activation functions. The result of the functions are further modified by the *pooling function*. The pooling function replaces the

result of a layer with a statistic summary of the output. The statistical method can vary based on the network design. For example, a typical method is max-pooling, where the largest value is reported from a rectangular neighborhood. The pooling function results in a layer output becoming approximately invariant to smaller translations to an input, meaning if input values are changed by a small amount, the resulting layer output will not change. For multiple tasks, pooling is essential for the CNN to be able to process input of varying sizes, and the function accomplishes this by varying the offset size between pooling regions. This results in, for example, a classification model that requires a fixed output dimension, to receive the correct dimensions regardless of input size, adding flexibility to the architecture design and input specifications.

The issue with the convolution operation and pooling function is that for each layer of the network, a dimensional reduction of the original input occurs. This limits the possible depth of the network, if the kernel is not very small, and also the spatial extent of the network.

To counteract the dimensional reduction, *zero-padding* is added to the input. Zero-padding means adding values of zeros to the outer rim of the input, making it wider, and in turn allowing control of the kernel width and the output size independently[GBC16].

The CNN-architecture has seen commercial success in several applications since its initial definition from Lecun et al.[LeC+89]. Since the ImageNet challenge in 2012, where Krizhevsky et al.[KSH12] designed a CNN architecture to classify high-resolution imagery, the usage of CNN architecture in image-recognition applications increased. CNN is known for its efficiency in image classification but is usable on other types of data that are not grid-like in form. In this report, for example, the use case for CNN is processing serialized time series data.

2.2 Machine Learning in Healthcare

ML and AI have increasingly become a component in healthcare for various applications, ranging from diagnostics to treatment planning. This section provides a contextual overview of the evolution and impact of ML and AI technologies in healthcare. The initial usage of the term "Artificial Intelligence" was at the Dartmouth Conference in 1956 and the term "Machine Learning" was coined shortly afterward by Arthur Samuel in 1959. The early attempts to incorporate ML concepts in healthcare would appear after only a few decades. In 1977, MYCIN, one of the first Expert Systems, systems that solve problems that human experts would, was a clinical consultation system that used reasoning techniques from the field of AI to assist physicians with therapy selection for patients with infections. Through interaction with physicians, it would consult with the physician, explain its reasoning during the consultation, and acquire new knowledge from experts without any experience in programming[EH77]. MYCIN was followed shortly by other expert systems such as Internist-I/QMR in the late 1970s and 1980s, developed at Pittsburgh University. These systems acted as a decision aid but could also critique physician evaluations and suggest laboratory tests. Using statistical AI methods such as ranking and partitioning algorithms, exclusion functions, and heuristic rules, they created ranked list outputs of diagnoses given inputted physician findings. Other expert systems, such as DXplain, Meditel, and Iliad followed in the 1980s and were considered to be the first CDSS. The Expert Systems were however rule-based systems that were built on predetermined IF-then statements based on expert knowledge and diagnostic conclusions. Therefore, they cannot be considered actual AI systems as we understand them today[Fra21]. During the 1990s, researchers began using novel ML techniques such as Artificial Neural Networks (ANN). ANNs gained prominence as a powerful ML technique, leading to various research-based systems in healthcare research. The capability of ANNs to learn complex patterns and their adaptability to various problems made them an attractive option for addressing healthcare challenges. One of the applications of ANN-based Computer-Aided Diagnosis (CAD) systems was in helping radiologists identify suspicious areas in mammograms, such as masses or microcalcifications, which could be indicative of cancer. For instance, in 1995, Lo et al. published a study demonstrating an ANN-based CAD system to detect microcalcifications in mammograms. The proposed system significantly improved detection rates compared to traditional methods, showcasing the potential of ANNs for cancer detection in mammography[Lo+95]. Another notable application of ANN in the 1990s was in diagnosis of cardiac disease. Baxt et al. conducted a study using an ANN to diagnose myocardial infarction in emergency department patients based on clinical data. The ANN-based system outperformed traditional statistical methods, indicating that ANN could enhance diagnostic accuracy in time-sensitive situations, such as emergency departments[G91]. However, these systems were only research-based and were not used in clinical settings. It would take until the 2010s that CDSS utilizing AI systems would be applied in the healthcare industry. Two examples of AI applications being applied in healthcare are IBM Watson for Oncology and IDx-DR. IBM Watson for Oncology is a

CDSS that utilizes natural language processing to analyze and extract information from unstructured data sources, such as medical literature, clinical guidelines, and patient records to create a personalized treatment plan for cancer patients that clinicians can consider[Som+17]. IDx-DR, another example of CDSS utilizing ML, used deep learning with a CNN to analyze retinal images to detect diabetic retinopathy, removing the need for human intervention streamlining diagnostic processes, and improving access to screening where experts are not available. The system was FDA-approved and CE-marked and has seen usage in the United States with positive reception[Abr+16]. The previous examples of Clinical Decision Support Systems (CDSS) with Machine Learning (ML) integration are just the beginning of a significant technological shift. The integration of ML technologies in healthcare is expected to improve various aspects of patient care, reduce costs, and enhance overall efficiency.

With the potential for personalized medicine, advanced diagnostics, acceleration of drug discovery, and cost savings, the global AI in the healthcare market is projected to experience substantial growth. Estimates suggest a compound annual growth rate (CAGR) of 37.5% due to increasing investments in AI research, technological advancements, and the growing demand for personalized medicine. This could result in a potential market size increase from 22.4 billion U.S. dollars in 2023 to a revenue forecast value of 208.2 billion U.S. dollars in 2030[Res21].

2.3 The Current Challenges of Machine Learning in Healthcare

The prospects for ML in a healthcare setting are many and revolutionary, with the promise of providing better care in every aspect of the industry. However, the integration of the technology is in its infancy, and several challenges in the healthcare industry limit the current scope of ML applications in juridical, cultural, technological, and privacy aspects. This section will focus on the technology and data privacy challenges currently in the healthcare industry that limit ML applications in the healthcare industry.

2.3.1 Data Privacy Aspect

The data processed in healthcare technology and systems are classified as highly sensitive data. The data contains personal information of individuals that, if misused, can put said individuals at risk for discrimination, identity theft, and misdiagnosis. This can result in distrust in patient-provider relationships. Therefore it is of high interest for the healthcare industry to recognize the sensitive data and protect it from potential risks of misuse.

Typical approaches to protect personal health data are anonymization and encryption through de-identification, stringent access controls to the data, and requirements to sign data use agreements that follow laws such as the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR) accordingly.

These approaches lead to more secure data handling and storage but also limit the amount of regulatory-approved data available. ML software needs massive amounts of data to train and achieve acceptable results, and if the amount of data is limited, the development becomes affected[Kay18].

Although the data is anonymized, re-identification of the data is still a possible issue. In a recent study, a group of researchers developed a generative copula-based method that estimates the likelihood of re-identifying an individual. The model concluded that with 15 demographic attributes, 99.8% of every American could be correctly re-identified from any data set[Luc19].

Re-identification doesn't require several attributes either. Another study had a set of 84 individuals between the ages of 34 to 89, who had undergone a Magnetic Resonance Imaging (MRI) scan 3 months before the study, where facial photos of each individual were taken. From the MRI images, three-dimensional and two-dimensional reconstructions were created from each respective individual. With publicly available facial-recognition software from Microsoft Azure, it was possible to correctly match 83% of the MRI scans and the photographs, and for 95% of the images, the correct photograph was one of the top 5 candidates[Sch+19].

The studies demonstrate that even the most heavily sampled de-identified data sets still don't meet modern standards for anonymization set by data privacy laws, such as the GDPR, and complicates the data handling step in ML development further.

2.3.2 Performance aspect

If an ML model shows significant promise in the development phase, with high prediction, validation and test performance, it is not guaranteed that the model will perform similarly in a final production setting. Commonly, the model underperforms significantly in the real-world setting compared to when tested. This phenomenon is called underspecification[DAm+20] and is a known issue in statistics. For machine learning development, there can be several high-performance models, with small arbitrary differences, such as different random seed or initial node values that demonstrate high prediction performance on the test data subset, but in a real-world setting will individually perform significantly different from each other. In their report, D'amour et al.[DAm+20], claim that underspecification is a ubiquitous phenomenon in modern machine learning development. In the paper, an experimental protocol in the form of stress tests was applied to several production-grade Deep learning pipelines. One of these pipelines was a Recurrent Neural Network model that predicted risk for acute kidney injury(AKI) based on electronic health records from the paper written by Tomašev et al.[N+19] The model was able to detect the onset of AKI 48 hours later with a precision of 55,8% in all episodes and 90, 2% in episodes associated with dialysis administration. However, the study found that there are operational factors, such as the time of day when blood tests measure creatinine values, and the number of blood tests taken that could affect performance. Two interventions were performed, where the time data were shifted with a fixed offset and blood tests removed not directly related to the diagnosis of AKI. This was further investigated

by measuring two different Long-Short Term Memory (LSTM) models. The interventions resulted in a worse performance of the models with larger dispersity, and the further investigated models showed substantial change in risk prediction after interventions with flipped predictions.

This puts pressure on the manufacturer of the ML application to further specify the requirements of the system, with rigorous testing with real-world data on several different models. Firstly it is a very time-consuming task, and secondly, the real-world data necessary for testing may not be readily available[Hea20]. This results in models that perform well in a research & development environment, but do not reach end production.

2.4 Machine Learning Operations

A development process that addresses the recurring issues of ML systems in development- and production environments is Machine Learning Operations(MLOps). MLOps is a combination of Development Operations(DevOps), which is an agile approach to software development- and operations management, Data Operations (DataOps), an agile approach for Database development and operations management, and Model Operations (ModelOps) which focuses on lifecycle management and governance of AI and ML models[BN22]. The definition of MLOps enunciated by the Continuous Development- Foundation(CDF) is as follows:
"The extension of the DevOps methodology to include Machine Learning and Data Science assets as first-class citizens within the DevOps ecology"[ME22].

MLOps is an approach to automate the end-to-end lifecycle of ML development, operations and monitoring. It aims to assist developers to establish cross-functioning teams that builds an application considering the data, the model, and the end application in small increments that are reproducible and reliable in short adaptation cycles. See figure 2.2 for a visual representation of an MLOps pipeline, from development to production.

MLOps has three major assets; Data, Model, and Code. In correspondence to these three assets, there are three major workflows that MLOps consist of; Data Engineering, Model Engineering, and Model Deployment.

Data Engineering

Data Engineering involves the acquisition of the data, using exploratory data analysis to obtain statistical and graphic information of the metadata, transformation, and feature engineering of the data into curated data sets, and splitting of the data for eventual model training into subsets of data.

The data is the center part of the MLOps lifecycle and prevalent in all phases of it, from model training, to -inference and -monitoring. The quality of the application is directly correlated to the data quality, requiring the data pipeline to be robust[23a].

Model Engineering

Model Engineering uses the acquired curated data to determine a proper model architecture for the task, the business inquiry, and the data structure. It involves the typical ML workflow, where the model is trained and validated on a training data set and a hold-out validation set respectively. The model is further evaluated with the test subset, and if the performance is approved, the model artifact is packaged for deployment, and the artifact is simultaneously saved in a model registry[23a].

Model Deployment

Model Deployment takes the trained, validated, and approved ML model artifact and readies it for inference in the application environment. It is firstly served to the application, and novel data is entered into the model, where model inference occurs and the resulting predictions, classifications, or generative results such as phrases or images, are used as part of the application. While the model operates it is monitored for its performance. This performance is evaluated to determine whether the model performs adequately, and metric-triggers assess whether the model performs well or a re-training is required. For each inference request, the results of the requests are logged for evaluation[23a].

Motivation

MLOps is a further development of already the successful development principles of agile and DevOps-lifecycles. DevOps is the most common development method in software development globally, where 47% of companies claim they use the methodology, stating that their products have a faster development cycle and reach the market faster. The development and operations have better security, ensure higher software quality and better communication between developer teams[Sta22].

MLOps practices differ however from DevOps practices. The Constant integration/Constant Deployment (CI/CD) practice differs as CI needs to test and validate further than code and component function, test and validate the data, evaluate the data schemas used for sorting and storing data, and the ML model. The CD is not about a single software service or package, but an entire automated system utilized for model training, validation, and prepared packaging into a model-prediction service[Tea23].

MLOps shows promise in the field of healthcare, both for the reasons stated above, but mainly for the potential of model monitoring and for integration of regulatory compliance in the development process. A case of utilizing MLOps practices to reach a certified medical device product is the development process of the medical device software Oravizio, the first CE-certified medical software that assesses patient's risks of a joint-replacement surgery, determining the risk for revision within 1 or 2 years after surgery or risk of mortality after surgery. The software development process of Oravizio demonstrates the possibility of having an automated development process while also being regulatory compliant[GT21].

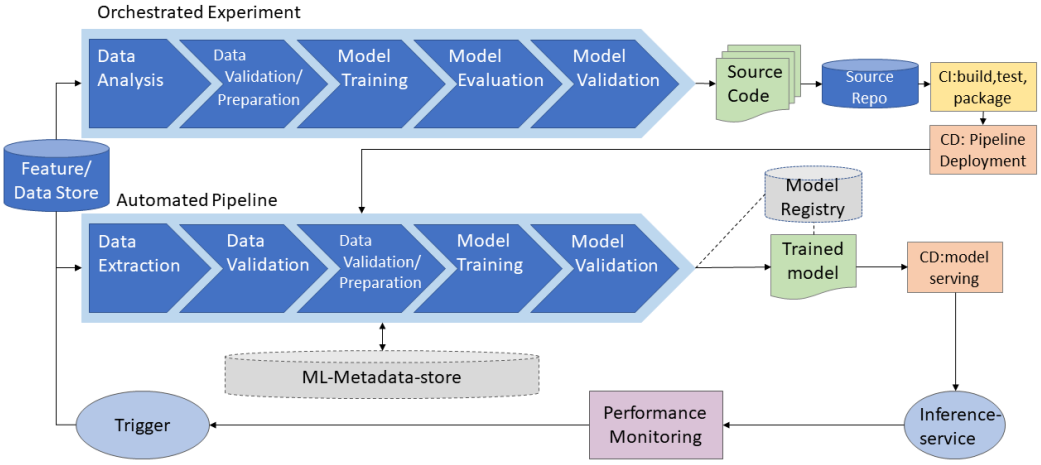

Figure 2.2: Example of an MLOps-pipeline with a CI/CD-implementation. The illustration is adapted from Google Cloud MLOps Guide[Tea23]. CI = Continuous integration, CD = Continuous Delivery.

2.5 Previous Research

The case study model the project uses for the MLOps framework is a neural network model developed by Rocheteau et al.[RLH21]. The model is designed to predict the length of stay and mortality in the ICU unit for data from the electronic health record, showing a methodological proof of concept for automated patient bed management. The model architecture is based on a CNN utilizing Temporal Pointwise Convolution.

2.5.1 Temporal Pointwise Convolutional Neural Network

Temporal Pointwise Convolution (TPC) is a combination of two subclasses of CNN, temporal convolution, and point-wise convolution. Temporal convolution convolves data over the time dimension and follows two key principles; no data leakage from the future and an output that is of the same length as the input, also in this case there is no weight sharing across features, it only occurs across each time point. Pointwise convolution, or 1x1 convolution, is typically used for dimension reduction in image processing. The main task of the pointwise convolution segment of the model is to extract information between features without information transfer over time. See figure 2.3 for a visual representation of the network architecture.

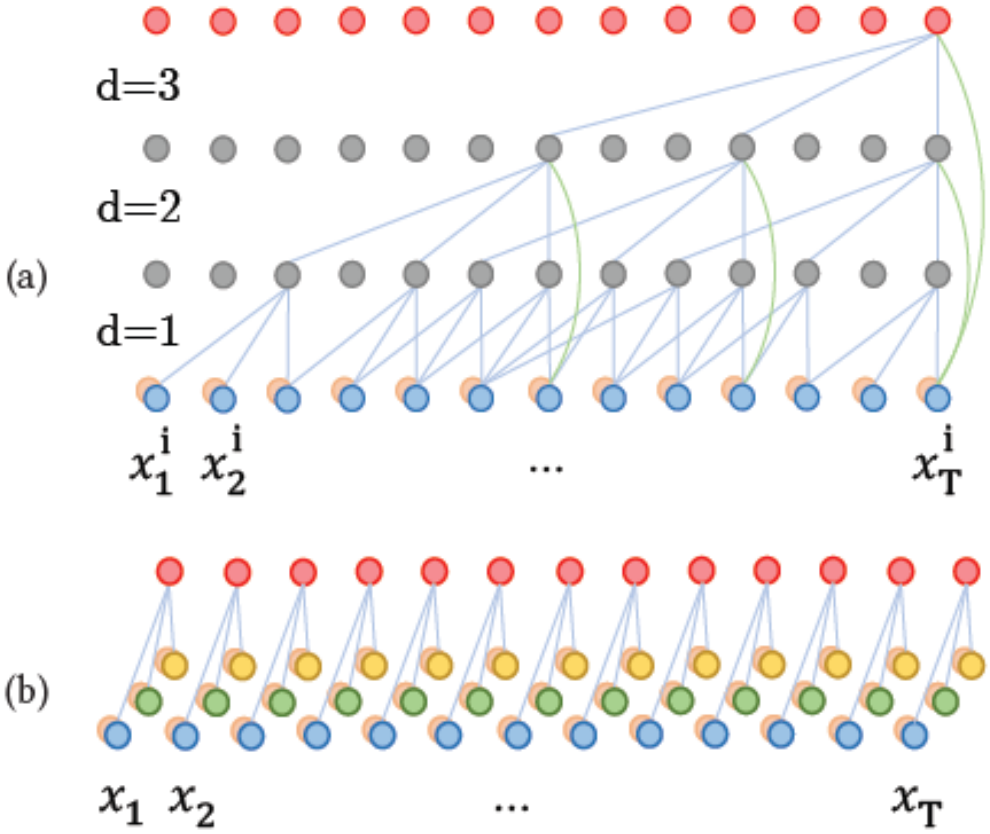

Figure 2.3: (a) Temporal convolution with skip connections (green lines). Each time series, i *(blue dots) and their decay indicators (pale orange dots) are processed with independent parameters. (b) Pointwise convolution. There is no information sharing across time, only across features (blue, green, and yellow dots). Illustration from Rocheteau et al.[RLH21]*

A TPC neural network combines the two former methods and runs them in parallel while adding skip connections to cope with sparse data, and removes the risk of input pollution by providing an anchor to the input.

The temporal section is first padded so the input- and output lengths are the same, batch normalization and dropout are made in the second layer, and skip connections are added in the third. The pointwise section first adds flat features and decay indicators repeated for the same length as the time series. Batch normalization and dropout are done for the second layer. The temporal and pointwise outputs are concatenated into the TPC-layer, which is used as input for a Rectified Linear Unit activation function. The output of the function is combined with static features and diagnosis embedding, and finally sent through two pointwise layers to obtain the final prediction with a hardtanh activation function that restricts the output if it is lower than 30 minutes or larger than 100 days See figure 2.4 and figure 2.5 for a visual illustration of the TPC network architecture.

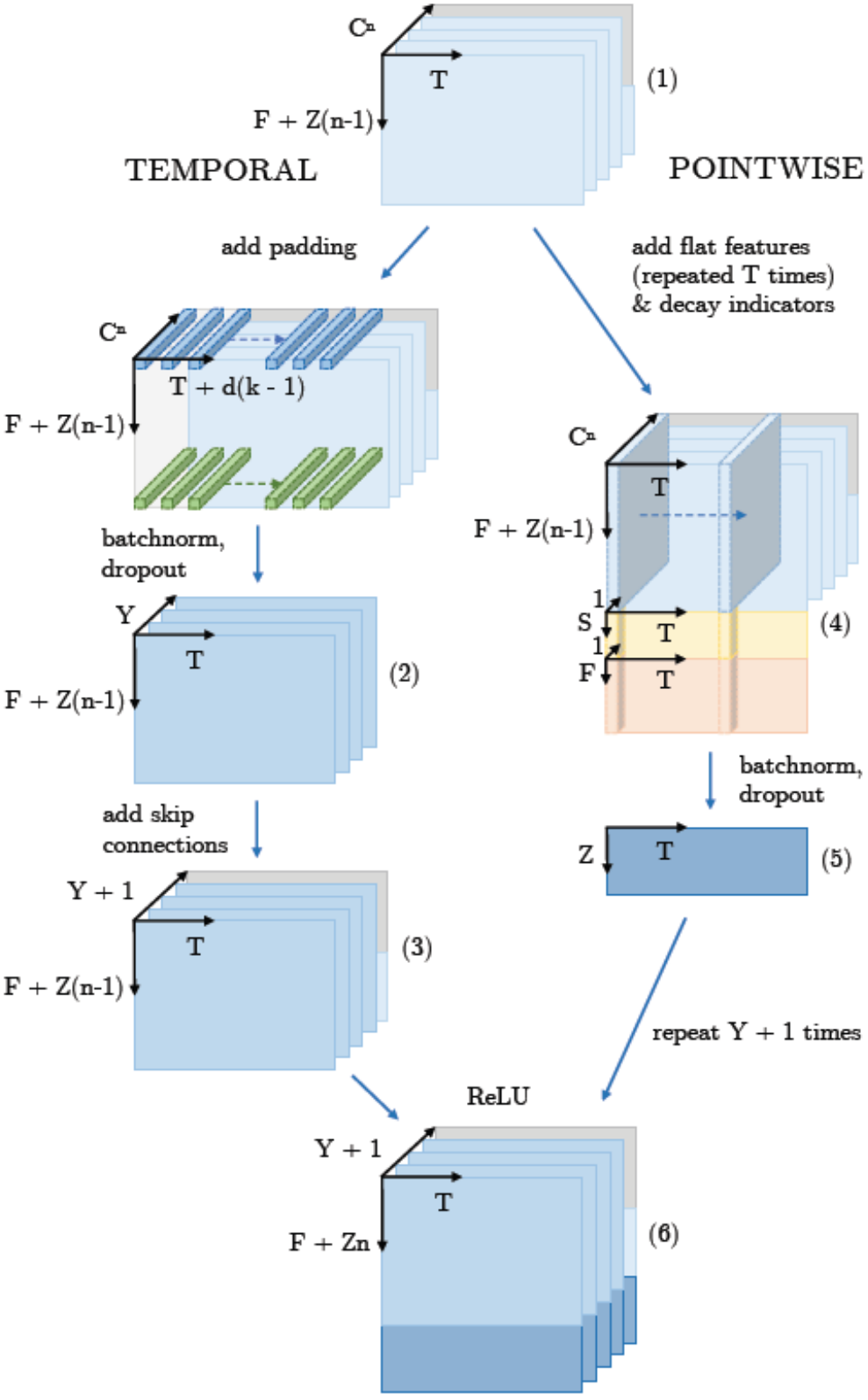

Figure 2.4: The n:th TPC layer. Left-sided padding (off-white) is added to the temporal side before each feature is processed independently. On the pointwise side, flat features (yellow) and decay indicators (orange) are added before each convolution. Illustration from Rocheteau et al.[RLH21]

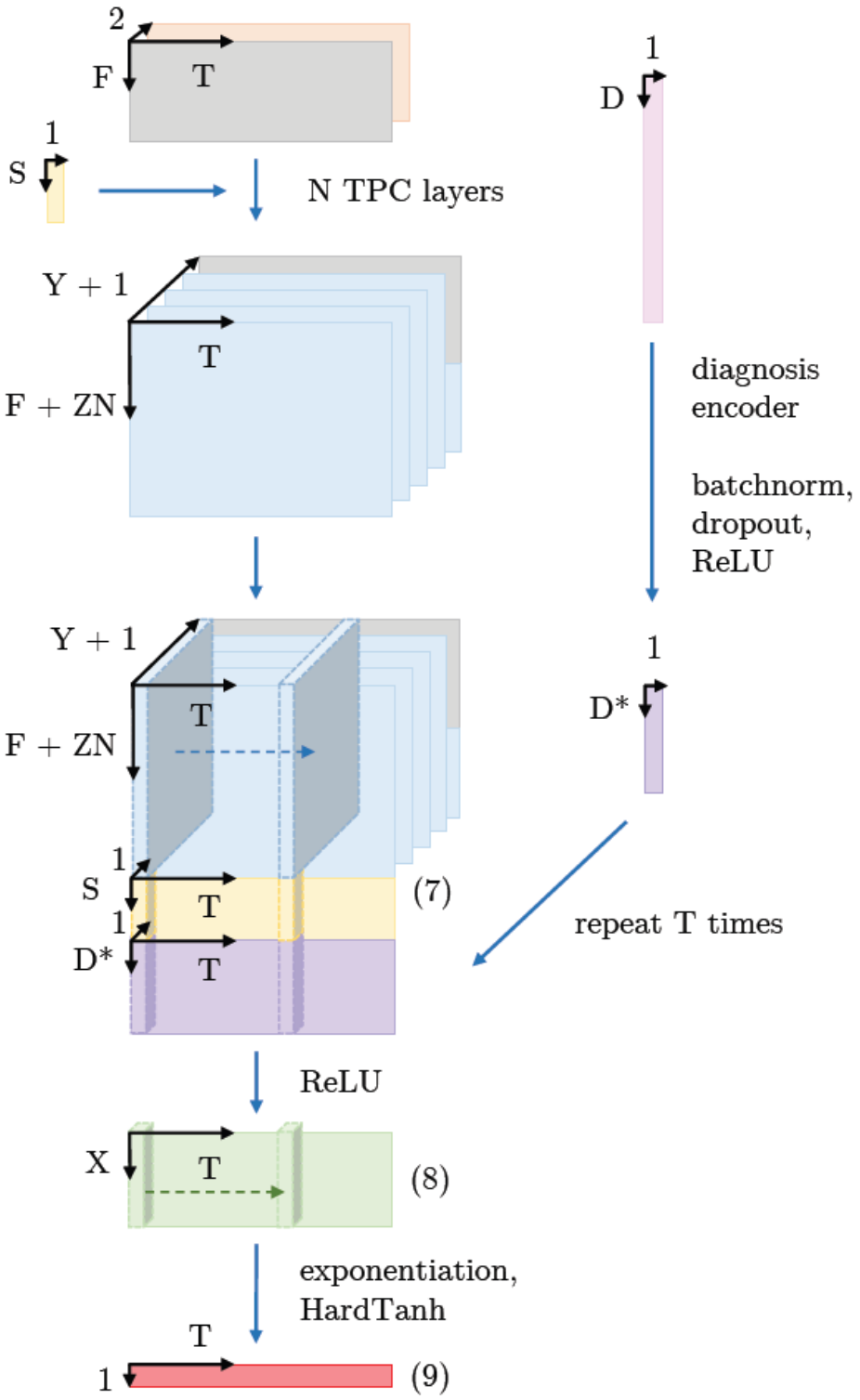

Figure 2.5: Further explanation of the TPC model. The original Time Series (grey) and the decay indicators (orange) are processed by N TPC layers before being combined with a diagnosis embedding D (purple) and static features S (yellow) along the feature axis. A two-layer pointwise convolution is applied to achieve final predictions (red). Illustration from Rocheteau et al.[RLH21]*

The Length of stay predictions that the model produces have no negative values making it positive-skewed. this makes the prediction task more chal-

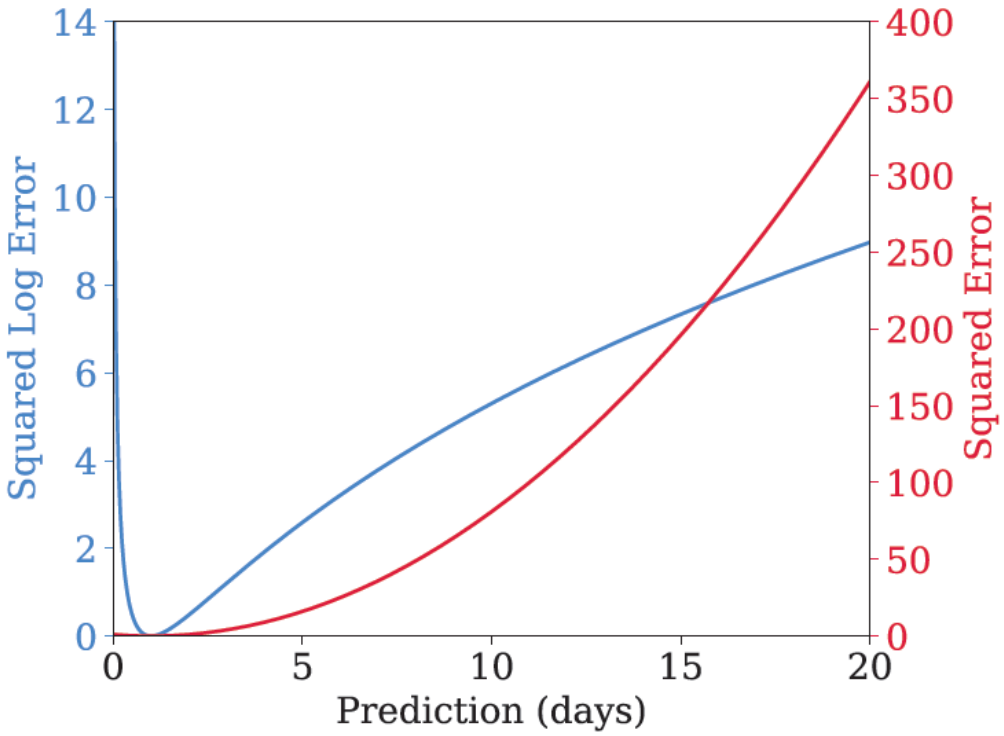

Figure 2.6: The behaviour of MSLE(blue) and MSE(red) when the true LoS is 1 day. Illustration from Rocheteau et.al.[RLH21]

lenging, and the training thereof. To address this, the appropriate loss function for the model is Mean Squared Logarithmic Error (MSLE), an alteration of the common Mean Square Error(MSE) loss function (2.7).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\log Y_i - \log \hat{Y}_i)^2 \tag{2.7}$$

MSLE is the loss function of choice as it penalizes proportional errors. Consider an error of 5 days for a 2 day-stay prediction versus a 30 day-stay prediction. If the MSE was used as the loss function instead of the MSLE, the 30-day-stay prediction would be strongly penalized resulting in the model predictions regressing towards the mean and becoming overtly cautious, which is inappropriate for bed-management prediction as over-predictions must not be too harshly penalized. Long-stay patients have a disproportionate effect on bed occupancy that needs to be taken into consideration. See figure 2.6 for a visual presentation of the behavior for the respective loss functions, where it is illustrated that the MSLE does not penalize large over-predictions compared to the MSE.

3

Methods

The objective of the project is to devise an implementation of model monitoring with the MLOps framework for CDSS, to determine if performance drift is detected. Specifically, a published ML model for a clinical task is trained and validated on a clinical data set. The data is initially split into two, separate subsets based on age groups to determine if performance drift occurs when the ML model is trained on one data set, but exposed to the other. This is done by utilizing MLOps practices and an MLOps environment. From the set objective, the method has been structured into subtasks to manage each specific task. The overall method is divided into the following subtasks:

- The choice of MLOps practices applied to the project, their function, and their usage. This is presented in section 3.1.
- The choice of the data set utilized throughout the project. This is presented in section 3.2
- The choice of resources and utilization of the resources from the works of Rocheteau et al. and A. Johnson et al. throughout the project. This is presented in section 3.3
- The choice of evaluation metrics to determine model performance after model training and during model monitoring. This is presented in section 3.4.
- Alteration method of the acquired data for model training and model inference respectively. This is described in section 3.5.
- Model training method, choice of hyperparameter settings, and interpretation of the training performance. This is presented in section 3.6
- Serving of the trained model for inference, how the performance of the model is monitored on the altered data, and how the model's performance is interpreted in comparison to the performance on the training data. This is described in section 3.7.

3.1 MLOps practices

MLOps practices are used for the case study to implement a software structure that supports model monitoring, and a clear division of tasks.

The practices used are adopted from the process suggested in the white paper "The Big Book of MLOps" written and published by Databricks Inc.[BN22]. See figure 3.1 for a visual representation of the development process utilized for the project. The development process covers all steps, from data acquisition to model deployment and -monitoring, that are taken for developing an ML model. Initially, data preparation occurs, which involves acquiring appropriate data and storing it appropriately. The initial step is followed by exploratory data analysis, where the acquired data is analyzed thoroughly to determine the potential ML-use case and the procedures to consider in the following step, feature engineering. Feature Engineering involves cleaning, transforming, or normalizing data into an appropriate shape or scale for the ML model to interpret the data correctly, based on the model's architecture. After feature engineering, Model Training is initiated, where a model of choice is tuned and optimized on the available training data. Afterwards, the model's training performance is validated in the Model Validation step. The model validation step involves further optimization of the model's performance on held-out validation data, tuning eventual hyper-parameters, and determining the appropriate training time. Finally, model performance is tested on test data that the model has not been exposed to, to determine if the model is fit for deployment based on its performance. The deployment step involves applying the model to the defined use case, where it is exposed to real-life data and the resulting model output is interpreted for decision-making. The model's performance and resulting output are monitored and evaluated during the monitoring step to ensure that performance is not worsening. If so, the development process is iterated from the appropriate step to update and further improve the model's performance. The development process has been utilized to structure the project method tasks presented in 3.

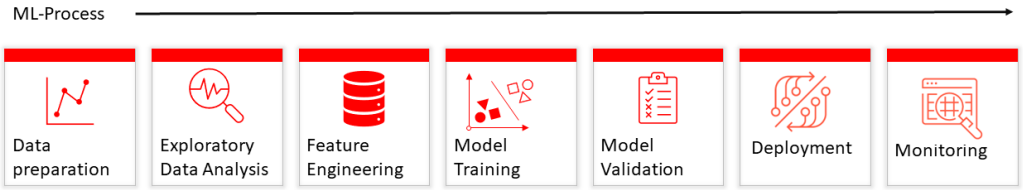

Figure 3.1: the MLOps model-development process, adopted from The Big Book of MLOps by Databricks Inc.[BN22].

From the "Big Book of MLOps" white paper, The case study implements the ML-deployment pattern "Deploy Models". A deployment pattern is a division of tasks into environments associated with certain parts of the ML model

development for ease of function. In the deployment pattern published and suggested by Databricks Inc., an ML-model artifact is generated in a development (dev) environment, that is then evaluated, validated, and approved in a staging (staging) environment before finally being deployed for inference in a production (prod) environment. The environments are representative of each step of the development cycle and the names of the environments are adopted from DevOps practices. See figure 3.2 for a visual representation of the "Deploy Models" deployment pattern. The dev environment is the environment for experimenting, augmenting, and generating code for the end application. In the case of ML, dev is the environment where the ML artifact and the associated code are trained, and developed, respectively. The staging environment is the environment for simulating the final environment, prod, by determining whether the applied software functions properly. For ML, it translates to determining whether the artifact performs as expected in a similar environment when ingested with data similar to the data it will be exposed to in prod. The prod environment is the final environment where the developed software is set up and used for commercial applications. For ML it means model inference on real-world data where the results are used for commercial applications, which can directly or indirectly interact with the software user. "Deploy Models" is the deployment pattern of choice due to the nature and scope of the project. The use of the MLOps practices, the development process, and the deployment patterns' usage will be further analyzed and evaluated in Section 5.

Figure 3.2: the MLOps "Deploy Models" deployment pattern, adapted from an illustration in The Big Book of MLOps by Databricks Inc.[BN22].

3.2 The MIMIC-IV data set

The data set used for model training, validation, and testing for the study is the Medical Information Mart for Intensive Care (MIMIC)-IV data set. MIMIC-IV is a data set where the data is originally from the MIMIC-III data set. MIMIC-IV provides critical care data from nearly 300,000 patients admitted to critical

units at Beth Israel Deaconess Medical Centre (BIDMC). The project is a collaboration with BIDMC and the Massachusetts Institute of Technology (MIT). The data is deidentified and patient identifiers have been removed according to the HIPAA. MIMIC-IV has further enhanced the original data through modulation, further distinguishing the data content for clarification. See figure 3.3 for a visual representation of the modularity[al23]. Further reading involving the management of the MIMIC-IV data can be found in chapter 7.

3.3 Usage of available resources

The resources and earlier works from A. Johnson et al.[Joh+18] and Rocheteau et al[Roc21]. are used throughout the project. From A. Johnson et al. the MIMIC code repository is used for creating readable data tables with the use of the structured query language (SQL) programming language. The MIMIC data are divided into three separate modules: *Hosp*, *ICU*, and *Note*. *Hosp* consists of lab tests, demographic and medication data admission/discharge/transfer records, and other hospital-wide data. *ICU* consists of data documented from the ICU bedside. The module contains data for charted observations of the patient, intravenous infusions done, patient outputs, and documentation of ongoing procedures. *Note* consists of discharge reports and radiology reports in free text form. the discharge reports are summaries of the patient's history and course over the entire hospitalization. Radiology reports are done for imaging studies, over a wide set of imaging modalities, such as X-rays, magnetic resonance imaging, and computed tomography. See figure 3.3 for a visual illustration of the modular structure.

The MIMIC-IV data that are processed and enhanced with the resources provided by Rocheteau et al[RLH21]. are stored in the *Hosp* and *ICU* modules because they contain data from general hospitalization and ICU. Therefore the Note module is not used for the project.

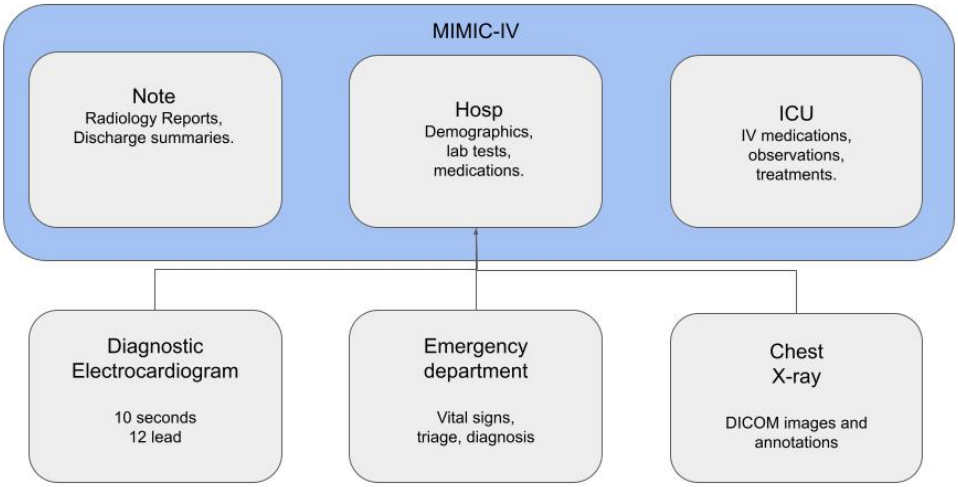

Figure 3.3: The modular structure of the MIMIC-IV dataset with examples of their respective content. The modules are linked with subject- and hospital-admission identifiers, and de-identified date and time.

The MIMIC data of interest are transferred and unpacked with the provided code to a local PostgreSQL server, where the *Hosp* and *ICU* modules are unpacked into their separate schemas and the stored data are sequenced into tables of varying size.

3.3.1 Featurization of the data

The MIMIC data stored in the PostgreSQL schemas are further edited with the use of a code repository provided by Rocheteau et al.

A new schema is created, and data from the original MIMIC schemas is copied and transferred into new tables stored in the new schema. This is done for the pre-processing scripts. From MIMIC-IV data, all patients over 18 years of age had a minimum recorded stay in the ICU of 5 hours and had at least one observation from a doctor. The pre-processing script, written in Python, makes for the MIMIC-data 2 sets of features. One set consists of static features and the other of time series. A total of 101 time series are extracted by the pre-processing scripts from the following tables: *lab, nursecharting, respiratorycharting, vitalperiodic and vitalaperiodic*. For time-series variables to be included they had to be present in at least 12,5% of patient-stay variables or 25% of lab variables. The lab variables were sparsely sampled in the data set. To deal with the missing data, first, the data is resampled into one-hour intervals, and then forward-filling over the gaps of the data is done. Forward filling is a method of data imputation that imputes missing values with the latest observed value. See figure 3.4 for a visual illustration of the forward fill-method. Rocheteau et al. argue that this is a more realistic approach to imputation than other methods, such as linear interpolation, for in a hospital setting, clinicians would only have the most recent value available[RLH21]. Data that is available before ICU admission is removed, and finally, decay indicators are added to the variables to specify where the data is becoming stale, using the decay value $0,75^j$, where j is the time since the last recording. In total, 12 static features were extracted from the *icustays, admissions, patients, and chartevents* tables. For both static features and time series, discrete and continuous variables were scaled to the interval $[-1, 1]$ with the 5th and 95th percentile as boundaries and the interval $[-4, 4]$ as an absolute cut-off to avoid large or incorrect inputs and avoid assumptions of the variable distribution. Categorical values were converted to numerical counterparts, where binary categories were coded as $[0, 1]$ and multi-categorical variables were one-hot encoded, which means a categorical value is translated into a one-hot value, consisting of a group of bits where only one bit is "hot", e.g. true. For example, a one-hot encoding with three states would be represented as [001, 010, 100][HH22].

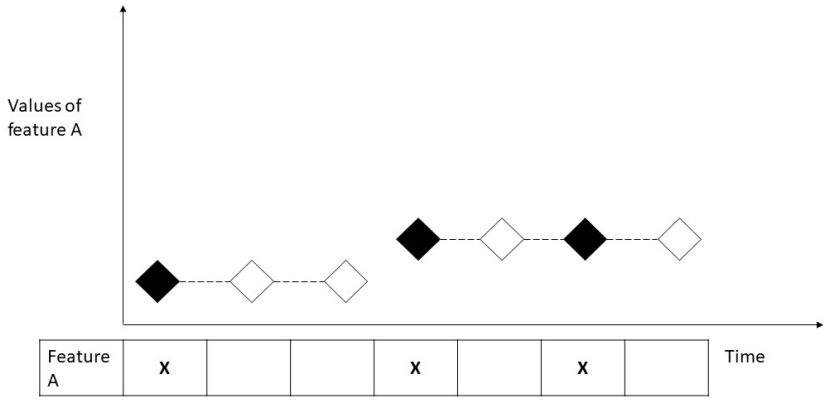

Figure 3.4: Illustration of the forward-filling method to impute missing data. Missing data is imputed with the last observed value.

3.3.2 MLflow

The model monitoring and maintenance are performed with the open-source library MLflow. MLflow is developed by Databricks Inc. for end-to-end development of MLOps workflows. The program has built-in integration with multiple libraries and can be used with any algorithm, library, or deployment tool[Mat18].

MLflow has four major components that appear and function differently for each partial stage in the end-to-end development:

- MLflow Tracking :
  MLflow Tracking is a component with an API and UI that logs parameters, code versions, metrics, and output files, that can in later stages be used for result visualization. It lets you log and query experiments in a Python API, Java API, R API, and REST API. The REST API will be used for this project. MLflow tracking is organized around the concept of *runs*, which are executions of some data science code. This can be a model prediction, metrics calculation, or a result visualization from predictions or metrics. It records information on the code version, the start and end time of a run, the source file that starts the run, Key-Value parameters, and metrics, where the metrics can be updated throughout the run like, for example, when one would plot the loss for each training-epoch. Runs also store MLflow artifacts, which are output files that can be in any format. The file can be a PNG image, an ML model, or a data file, that can be accessed or visualized from the API and UI. Runs are by default recorded locally, but can also be stored in a database, an HTTP server, or a Databricks workspace[23f].

- MLflow Models:
  MLflow models enable us to package a large variety of ML models from different types of libraries and deploy these models to a variety of model-serving or inference platforms. An MLflow model is a packaged directory of different arbitrary files that contain an MLmodel file that defines multiple *flavors* the model can be viewed in. Flavors are the main feature of MLflow Models that makes the model format and code interpretable, without needing tools that the model is planned to work with integrated for that specific code library. The package defines several standard flavors from different kinds of ML code libraries that the built-in development tools all support[23d]. With the MLflow Models component, a logged or registered model can be deployed, or "served", to a port or a server of choice for model inference and putting the model into a production environment. Through the MLflow Models API, the served model can be given batches of data for predictions.

- MLflow Model Registry:
  The MLflow model registry is a component that stores models and tags them with their development stage. With the use of a set of APIs and UI, the model registry is used to manage the lifecycle of an MLflow model. The Model registry uses concepts that facilitate the model lifecycle and uses earlier component-specifics for that purpose. An MLflow model that is created from an experiment or run is logged with the model flavor's logging function.

  ```
  mlflow.<model_flavor>.log_model()
  ```

  Once logged the model can then be registered. A registered model has a unique name, a version number, and an associated transitional stage[23c].

- MLflow Projects:
  MLflow Projects is a packaging format for data science code to make it reusable and reproducible. It is primarily a convention for code organization and -documentation for experiment reproducibility both manually and for automated tools. The project is a directory or a file that contains the model code and associated files that contain environment specifications and library dependencies that are downloaded and run before the production of the code[23e].

MLflow is the ML lifecycle manager tool of choice in the use case of this project. The functions and methods in the software library give us a structured visualization of the performance of the trained models, and efficient tools for model deployment that replicates a similar inference environment for the model's use case, where a clinician would be giving the model available patient data for a length-of-stay prediction. From the UI and API interactions, MLflow gives us a built-in monitoring tool for performance metrics. The utilization of the software library will be further analyzed and evaluated in Section 5.

3.4 Evaluation Metrics

The Length-of-Stay prediction from the TPC model is a numerical value, and the most appropriate metrics for such predictions are metrics used for regression problems[GK22]. Additionally, the evaluation metrics specified for the model in the work of Rocheteau et al. use regression metrics for the Length-of-Stay problem[RLH21].

The metrics of choice are chosen based on the metrics used for the model in the original paper, and their interpretability:

- Mean Absolute Deviation (MAD)
  MAD is a method for determining the deviation from the mean data value. it is a common metric to determine the eligibility of a forecast prediction. As the performance is based on the deviation disparity, the smaller the MAD measured, the better the performance

$$\text{MAD} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \bar{y}|, y \in \{y_1, y_2, \ldots, y_n\}$$

- Mean Absolute Percentage Error (MAPE)
  MAPE is a method to determine the percentage deviation of the predicted value from the real prediction value. it is calculated using the absolute error of the prediction divided by the true value and multiplied by the percentage factor of the set. The performance of the MAPE is deemed better the smaller the metric is.

  The formula for the MAPE utilized in the project is as follows:

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{\max\left(\frac{4}{24}, y_i\right)}\right|, y_t \in \{y_1, y_2, \ldots, y_n\}$$

  The denominator in the MAPE equation is the same as in Appendix D in the paper by Rocheteau et.al. to manage local unbound values[RLH21]. The modification is to be interpreted as if the duration of the LoS is smaller than 4 hours, the value is set to $\frac{4}{24}$, to address values too close to zero.

- Mean Squared Error (MSE)
  MSE in regression prediction measures the squared difference of the residual value. It is interpreted as the lower the value, the better the metric performance.

  The MSE formula is defined as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, y \in \{y_1, y_2, \ldots, y_n\}$$

- Mean Squared Log Error (MSLE)
  MSLE is an altered error method from the MSE, that adds the log function to the error summation. This reduces the large outlier values and

reduces their impact on the resulting error margin, which gives a more balanced impact on the error margin for all values. The performance of the MSLE is deemed better the smaller the metric is. The MSLE formula is defined as follows:

$$\text{MSLE} = \frac{1}{n}\sum_{i=1}^{n}(log(1+y_i) - log(1+\hat{y}_i))^2, y \in \{y_1, y_2, \ldots, y_n\}$$

- Coefficient of Determination(R^2)
  R^2 is an indication of the proportion of variation in the prediction values that can be explained by the independent variables of the regression model. The metric is in the range of $[0, 1]$, and the larger the metric value, the better the variance explained by the model, i.e. the "goodness-of-fit" of the model, meaning that if the R^2-score is 1, the variance is fully explainable by the independent variables, and 0 means their variance is not explained by the independent variables. The formula is as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, y \in \{y_1, y_2, \ldots, y_n\}$$

  The numerator is the sum of all residuals squared, while the denominator is the sum of squares total.

3.5 Data Alteration

As the focus of the project is the monitoring of deployed ML models, it is important to detect an apparent change in the model performance. For the model to perform differently, the MIMIC data has been split and altered into two data sets, where one is for training, validation, and testing purposes and the other for inference. The split factor of choice is the retirement age of the United States population. At older ages, the multitude of physiological changes that occur in the bodies of elderly people can affect the prediction result[BS81]. The training and validation set consists of patient data with patients in the age span of 18 - 66, while the inference data set has patient data with patients aged 67 and older. The MIMIC data set is based on patients in the United States. Citizens of the U.S. gain full social security benefits if retiring at the age of 66-67 depending on one's decade of birth, and access to the government Medicare insurance package when turning 65. While the normal retirement age in the U.S. is 64,9 and 64,7 for men and women respectively, the Non-retiree's retirement age target has increased beyond age 66, meaning if the targets are met, the normal retirement age will see an increase in the future[Jon22]. See figure 3.5 for an illustration of the data alteration and the resulting subsets from the alteration.

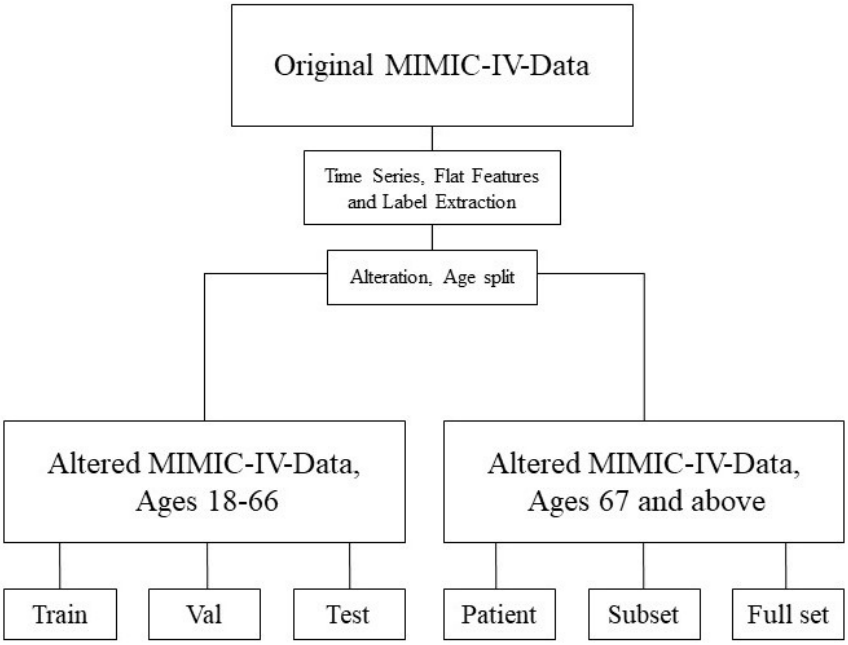

Figure 3.5: Illustration of the workflow for the pre-processing steps of the data, the eventual age split, and the separation of training, test, and validation for the 18-66 data, and unique patient, subset, and full set of the 67-and-above data.

3.6 Model Training

A TPC model is trained on the pre-processed MIMIC-IV data that contains patient data with an age span between 18 and 66. This is done locally in the Python programming language, and the trained model is then registered in the MLflow framework.

The hyperparameters of the TPC model are set based on the table values presented in the report by Rocheteau et al.[RLH21]. The parameters of choice are set for the MIMIC-IV data set, that acquired the best performance result presented from their work. The prediction task of focus is the Length of Stay of the patient in the ICU, and the mortality prediction task will not be utilized[RLH21].

TPC-Specific	
Temporal Specific	**Pointwise Specific**
Temporal Channels: 11	Pointwise Channels: 5
Temporal Dropout 0.05	Main Dropout*: 0
Kernel Size: 5	
TPC Common	
Batch Normalisation*: True	
No. TPC Layers: 8	
Non-TPC-Specific	**Global Parameters**
Main Dropout*: 0	Batch Size: 8
Final FC Layer Size*: 36	Learning Rate: 0.00221
Batch Normalisation*: True	

Table 3.1: Best Hyper Parameters for the TPC-Network with the MIMIC-IV data for LoS prediction. From Rocheteau et al.[RLH21].

The performance of the trained model is evaluated with the evaluation metrics described in section 3.4 with the test set from the data. The values calculated from the metrics are used as the valid baseline performance when evaluating the model prediction performance on the inference data. A performance variation is deemed significant if larger than a 5% error margin.

3.7 Model Inference

Using the MLflow component MLflow Models, the trained TPC-model, now logged and registered, is served to a local server port for model inference. The model will be ingested with the altered MIMIC-IV data, with patients of ages 67 and older. The resulting prediction data from the model is saved and evaluated with the metrics mentioned in section 3.4, and the metrics' performance will be compared to the baseline determined with the test-set from the data containing patients in the ages of 18 to 66.

4

Results

The results section showcases the results for certain subtasks of the method subtasks. The results are presented in the following way:

- The resulting data distribution after pre-processing and the data alteration. This is presented in section 4.1.
- The resulting feature shape of the data, the difference from the original MIMIC-IV data set, and how the data alteration affected the final data. This is described in section 4.2.
- Illustrations of the result from the TPC-model training, with the calculated evaluation metrics, and graphs showcasing the performance of the model on the training data set. This is presented in section 4.3
- Illustration of the result from TPC-model inference, with similar performance metrics and illustrative graphs as in section 4.3 to showcase the performance of the TPC-model on the inference data set, and the performance variation compared to the training- and validation performance. This is presented and described in section 4.4.

4.1 Distribution of the MIMIC-IV Data

The resulting distribution, after pre-processing, of the MIMIC-IV data, is showcased in table 4.1. The splitting of the data based on age resulted in an approximate 50/50-split of the data, both patient-wise and time-series point-wise. The data consisting of patients in the age span of 18-66 were used as the training data for the TPC model, and after pre-processing were split further into training, validation, and test subsets.

The other half of the data, consisting of patients in the age group 67 and older, are used for the model inference. The data are not split but instead merged into a larger data set that the trained TPC model, after being served, performs predictions on for model inference, and the performance is monitored.

From table 4.2, we can see the mean age gap between the two data sets, and that there is a notable age difference of approximately 27 years between the data sets. However, The LoS of the two patient groups are very similar, with a minuscule time difference when comparing the mean LoS (0.01 days, 14.4 minutes), and a minor time difference when comparing the median (0.15 days, 3.6 hours)

MIMIC-IV	
Total Patients: 299,712	
Patients admitted to the ICU: 73172	
ICU Patient Distribution Post-Processing	
Training Data(age 18-66)	**Inference Data(Age 67+)**
Total patients: 36900 (50,43%)	Total Patients: 36272 (49,57%)
Training Set: 26032	Merged
Validation Set: 5473	Merged
Test Set: 5395	Merged
Mean Age(years): 51.32	Mean Age(years): 78.26
Mean LoS(days): 3.45	Mean LoS(days): 3.46
Median Los(days): 1.85	Median LoS(days): 2.00
Time-Series-Points Distribution Post-Processing	
Training Data(age 18-66)	**Inference Data(Age 67+)**
Total: 2748326	Total: 2751119
Training Set: 1939110	Merged
Validation Set: 403881	Merged
Test Set: 405335	Merged

Table 4.1: Distribution of patients, their mean age, mean and median Length of Stay(LoS) in the ICU, and distributions of time-series-points in the altered MIMIC-IV Data sets.

4.2 Data Alteration

Before the pre-processing of the MIMIC-IV data is made to match the expected input shape for the TPC model, the data is split into two separa te data sets, where the data used for training purposes consist of patients in the age span of 18 to 66 and the data for inference purposes consist of patients in the age

span of 67 and older.

After pre-processing, a set of flat features in both data sets was excluded from each respective feature file. Feature exclusion is determined based on the occurrence of the feature data for each patient. The exclusion rule is if the feature is not measured or present for more than 1000 patients in the data set, then the feature is excluded from the final, pre-processed flat feature file.

The flat features that were excluded after pre-processing, what data set they were removed from, and the type of data they represented, are showcased in table 4.2.

Table 4.2: Flat features excluded from the altered flat feature data sets

Feature:	**Type:**	**Excluded from:**
Admission location transfer from skilled nursing facility	Categorical(Boolean)	Both
Admission Location Walk-in/Self-Referral	Categorical(Boolean)	Both
First Care Unit Neuro-Intermediate	Categorical(Boolean)	Inference set
First Care Unit Neuro-Stepdown	Categorical(Boolean)	Both
First Care Unit Neuro-surgical Intensive Care Unit (Neuro SICU)	Categorical(Boolean)	Both
Insurance MedicAid	Categorical(Boolean)	Inference set
Race, White-Other, European	Categorical(Boolean)	Both
Insurance Misc.	Categorical(Boolean)	Training set

To make the input features have the same dimensions, all the features in table 4.2 were excluded from both the training- and inference flat features.

4.3 Model Training

Table 4.3 contains the evaluation metrics of the result from the test set from the 18-66 data set. These metric results are used as the baseline for determining drift in the inference performance.

μ	σ	MAD	MSE	MAPE	MSLE	R^2
4.067	4.893	2.401	27.404	61.089	0.598	0.459

Table 4.3: TPC-network performance on the test-set of the patient data within the age-span 18 to 66.

The model reaches the best performance for the training data set after 10 epochs, and the best performance for the validation data set also after 10

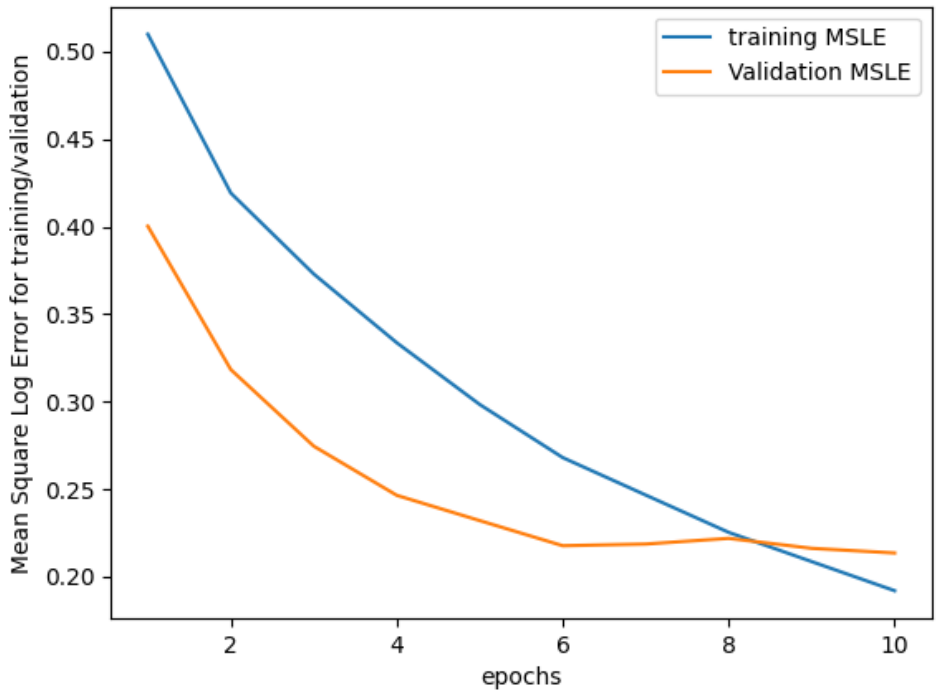

Figure 4.1: Learning curve of the loss for each epoch for the training- and validation data sets. The loss function utilized was Mean Square Log Error(MSLE).

epochs. At epoch 8, the loss of the performance for the training- and validation data is very similar, with the training loss being 0.2254 and the validation loss 0.2219. See figure 4.1 where the training loss curve, and validation loss curve, are respectively displayed.

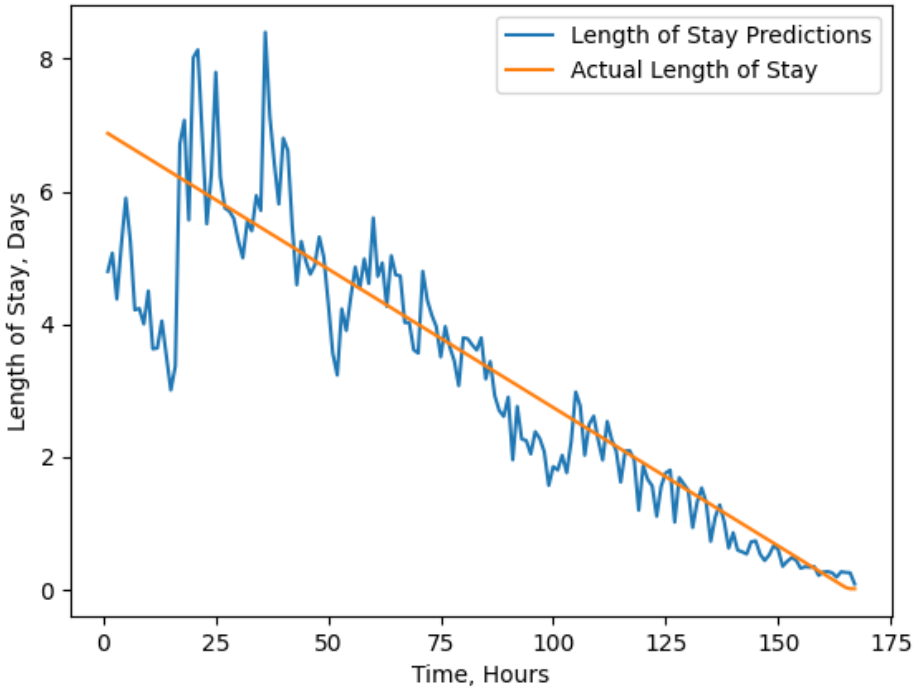

Figure 4.2: Plot of the model's predicted Length of Stay and the actual length of stay for a unique patient in the age group of 18 to 66.

The time series of a unique patient is displayed in figure 4.2. The figure illustrates the predicted LoS and the actual LoS of one patient, from the patient group within the age span of 18 to 66. The predictions (blue) are made in 1-hour periods, where the model predicts the LoS of a patient for each hour. The actual LoS (orange) is illustrated as a linear-receding line where the initial value is the total length of stay, and the line recesses towards 0 in a 1-hour time scale. For each hour, the length-of-stay prediction for the patient is illustrated.

Figure 4.3 and 4.4 illustrates, beyond the patient in 4.2, 11 more unique patients with their length-of-stay prediction and actual length-of-stay co-plotted.

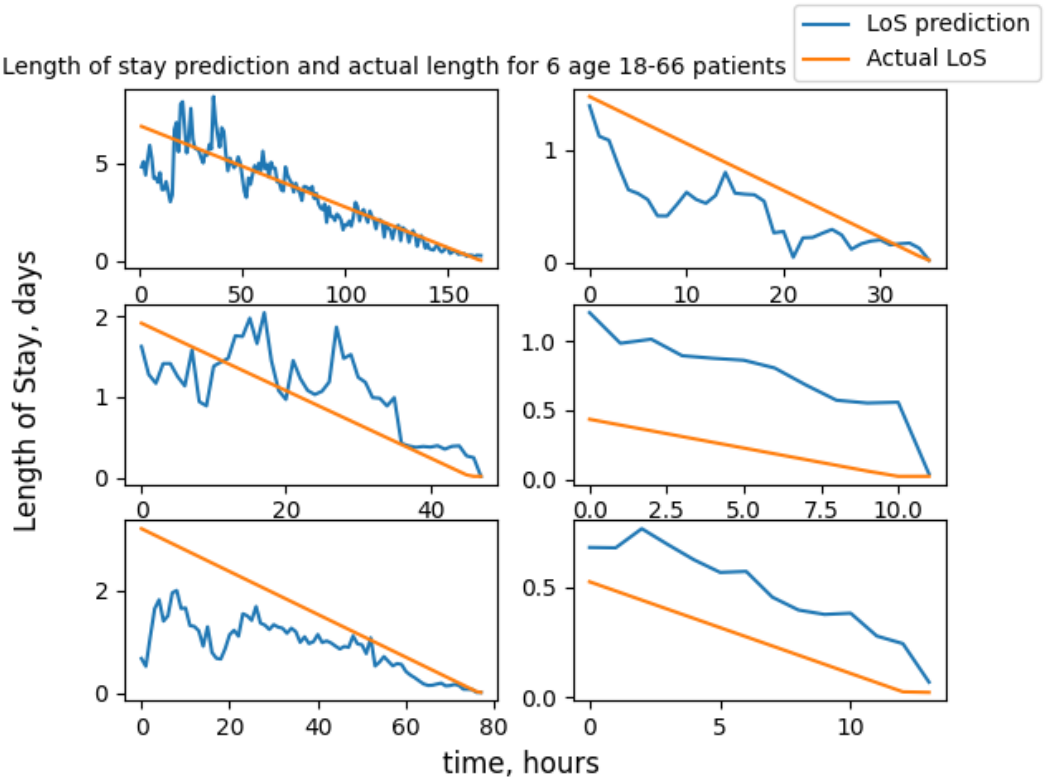

Figure 4.3: Length of stay prediction and actual length of stay for 6 patients in the age group of 18 to 66.

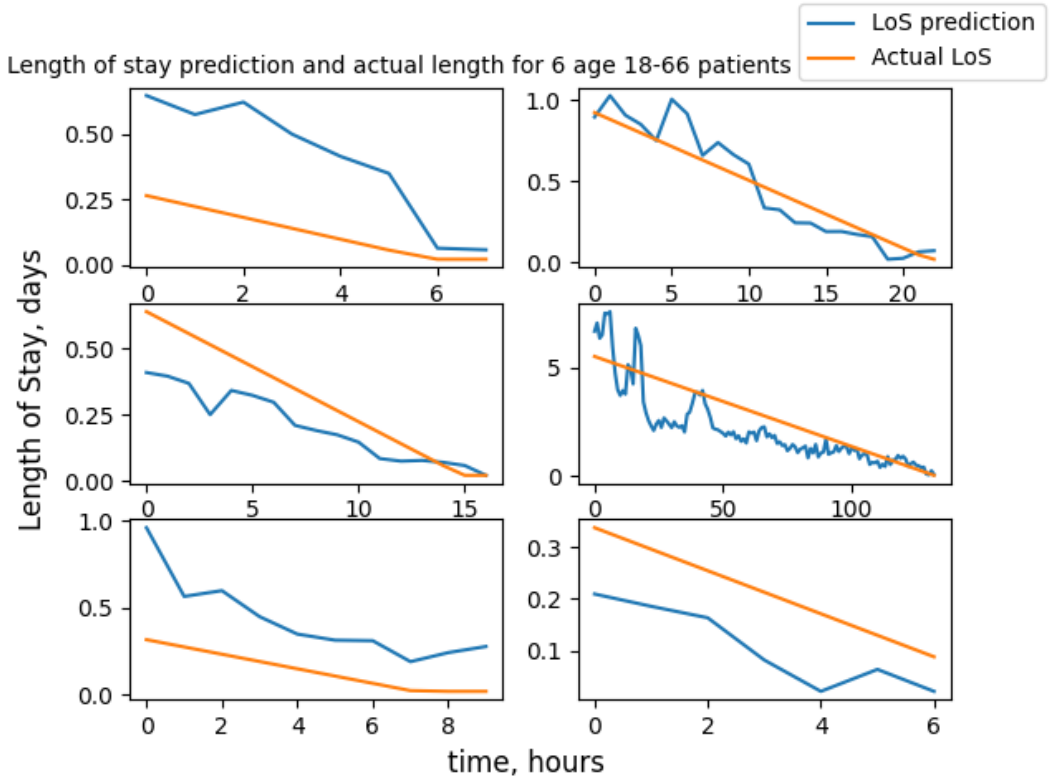

Figure 4.4: Length of stay prediction and actual length of stay for 6 patients in the age group of 18 to 66.

For further clarification of the prediction distribution and accuracy, a histogram of the residual prediction values was made. Figure 4.5 illustrates the residual histogram of the test set from the training data, showcasing the quantitative distribution of the residuals, with the residual values on the x-axis, and quantity on the right y-axis. On top of the histogram, a kernel density estimate of the residuals is co-plotted, illustrating the percentage distribution of the residuals, with the values shown on the left y-axis.

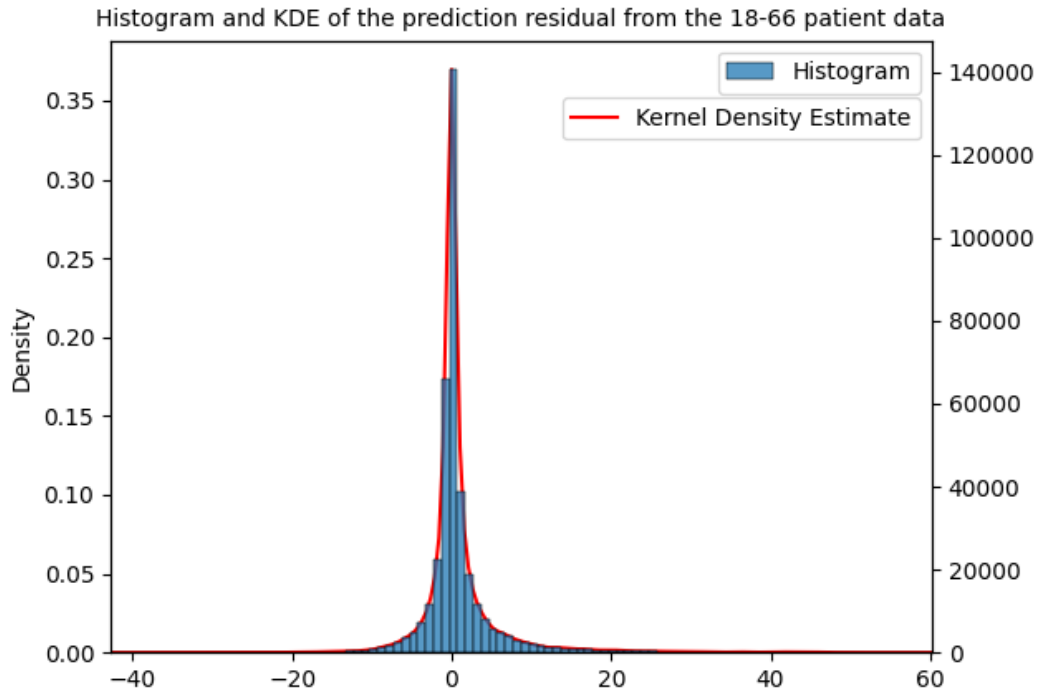

Figure 4.5: Histogram and Kernel Density Estimate-plot of the residual values of the inference prediction on the 18-66 data.

4.4 Model Inference

The results showcased in this section are from monitoring of the model predictions when it predicts the inference data. The performance when the model processes the inference data is showcased in table 4.4. The performance is determined on the entire inference dataset. All the patients and data points quantified in the inference data showcased in 4.1 are used for calculating the performance. As showcased in table 4.4, compared to the baseline performance from 4.3, The model shows a significant drop in performance. The most severe drops in performance are for the MSLE and MAPE metrics, with a performance loss of 44.64% and 41.13% respectively. The R^2-coefficient was reduced by 29.85% from the baseline, meaning that the variance is less explainable by the independent variable features. The MSE metric had a 17.93% reduction from the baseline metric, showcasing also a significant performance loss. The MAD metric had the lowest reduction from the baseline metric with a 6.08% reduction, which is a significant performance loss if applying a 5% error margin from the baseline metrics. The mean prediction of LoS on the patient data is very accurate when compared to the mean LoS of patients in the age group 67 and older in table 4.1.

μ	σ	MAD	MSE	MAPE	MSLE	R^2
3.450	4.250	2.547	32.318	83.316	0.865	0.322
performance loss	(%)	6.08%	17.93%	41.13%	44.64%	29.85%

Table 4.4: TPC-network performance on the test set of the patient data within the age-span 67 and older.

When comparing the figures illustrating the predictions of the unique patient from group 18-66 in figure 4.2 and the unique patient from group 67 and older in figure 4.6, the predictions in the latter figure has a larger variance and residual. This can be observed when comparing the Y-axes (The LoS in days) of the two figures. It is of note however that the LoS of the patient in figure 4.2 is longer than for the patient in figure 4.6. This is also not a significant result for the entire performance, as it is only one patient. However, similar results are showcased in the figures 4.7 and 4.8, which are group plots of a sample size of all patients from the patient group 67 and older, when compared to the group plots 4.3 and 4.4, of the same sample size. The larger variance and residual can be observed when comparing the Y-axes of the figures.

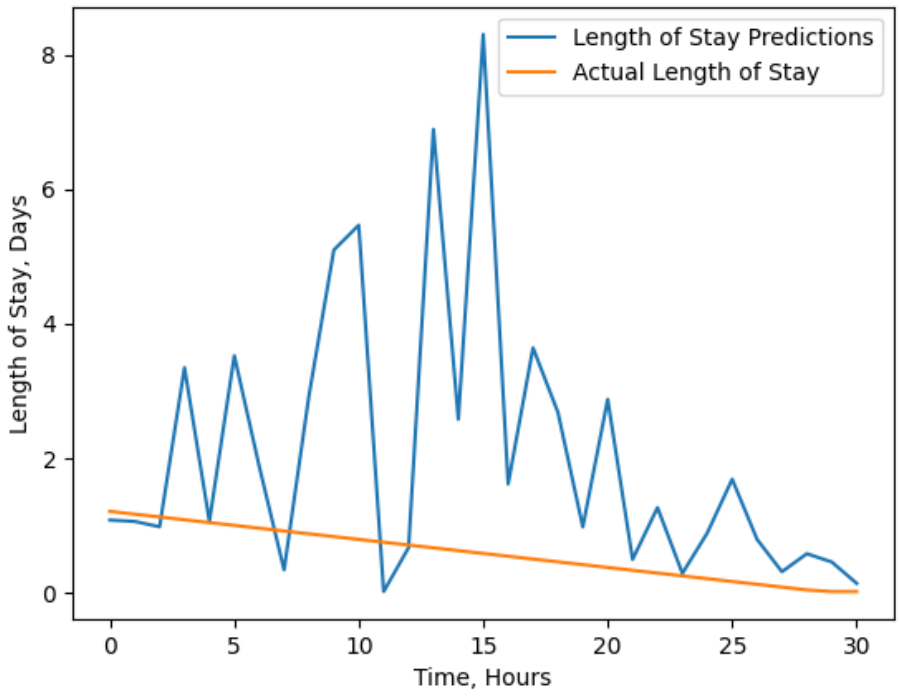

Figure 4.6: Plot of the model's predicted Length of Stay and the actual length of stay for a unique patient in the age group of 67 and older.

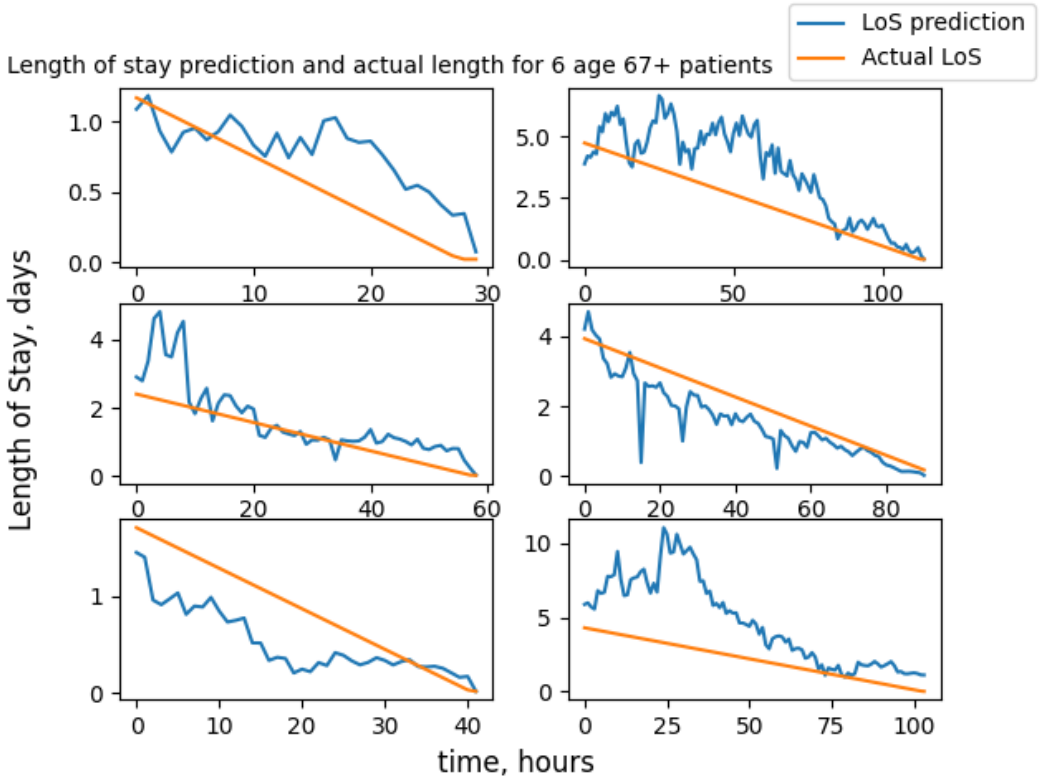

Figure 4.7: Length of stay prediction and actual length of stay for 6 patients in the age group of 67 and older.

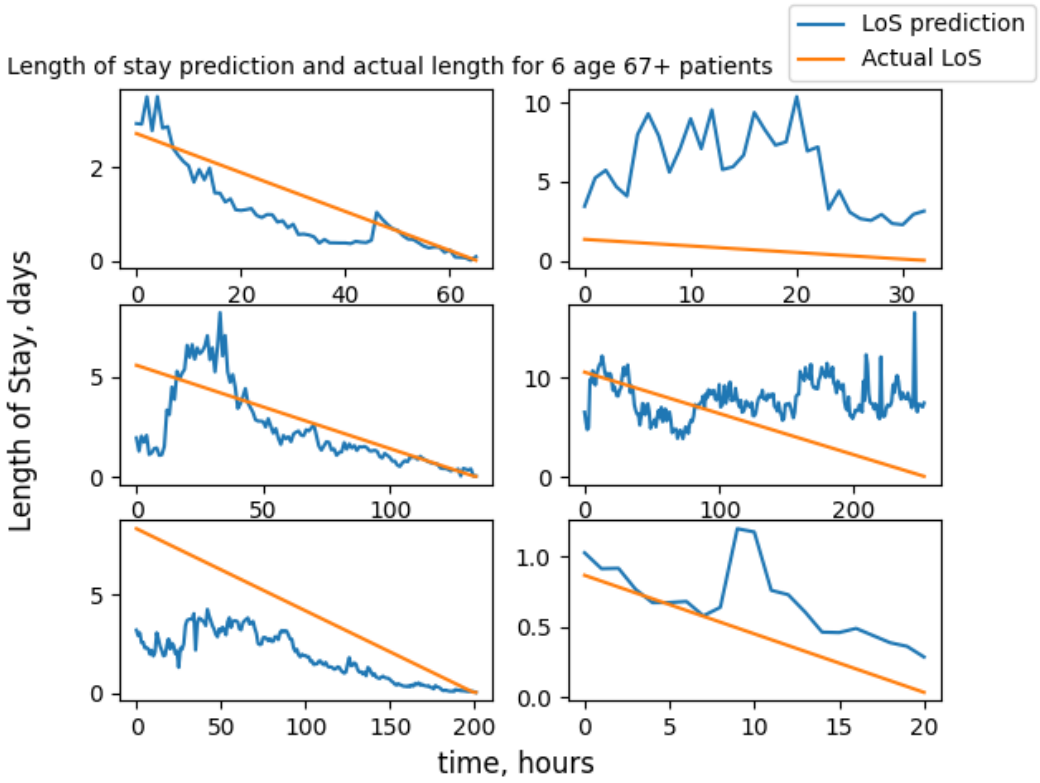

Figure 4.8: Length of stay prediction and actual length of stay for 6 patients in the age group of 67 and older.

The histogram and kernel density estimate plot from figure 4.9, in comparison to the similar plots in figure 4.5, have a smaller quantity and density of residual values surrounding zero. Figure 4.9 also showcases a larger spread of outlying residual values, and a larger positive skew from zero and outwards. The histogram is generated from a subset of the 67-and older patient data that contains the same amount of data points as the histogram in figure 4.9.

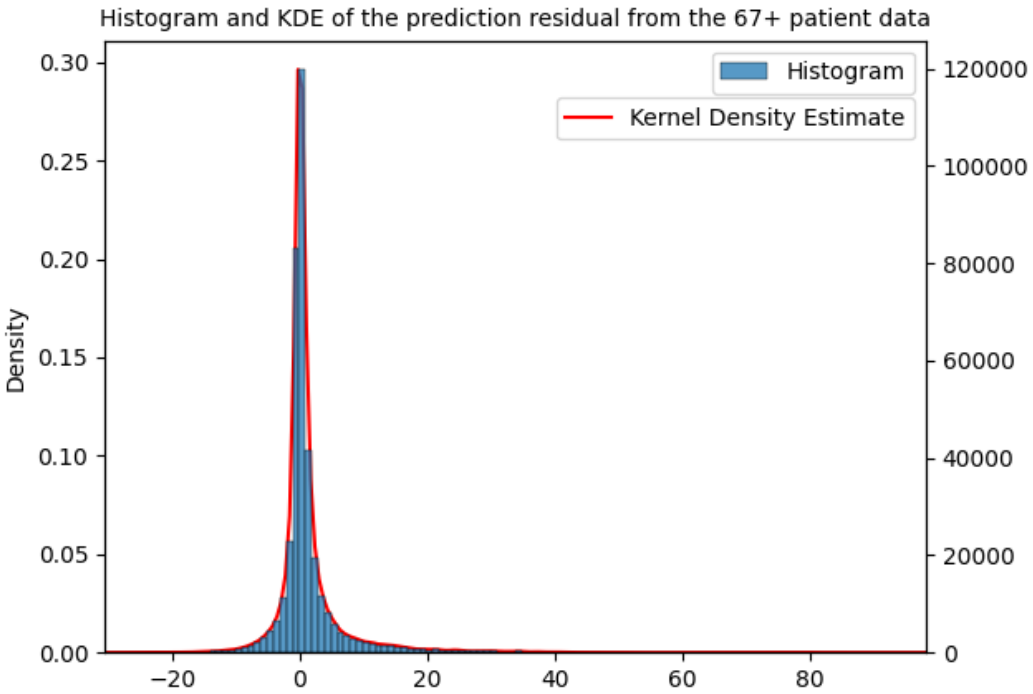

Figure 4.9: Histogram and Kernel Density Estimate-plot of the residual values from a subset of the inference prediction on the 67+ data. a subset of values are chosen to match the residual-value set size in figure 4.5.

5

Discussion

5.1 The MLOps Practices

The development process suggested by Databricks Inc. and illustrated in 3.1 was utilized as the structure for the method tasks. The data acquisition, -preparation, and exploratory data analysis steps covered the MIMIC-IV data acquisition and analysis. Based on the data, the TPC model became the choice of use case model. For the project's purpose, the monitoring and detection of performance drift, the data set was featurized and split into age-demographic subsets. The model training and validation steps were followed similarly as described in section 3.1, utilizing the optimal hyper-parameters presented in the report by Rocheteau et.al.[RLH21]. The serving of the trained TPC model, ingestion of data into the model, and evaluation of the model's performance followed the structure of the deployment and monitoring steps. The resulting utilization of the development process structure was beneficial for the project as it structured and segmented the necessary tasks appropriately with a data-centric approach in mind. The nature of the project not being iterative made it difficult to determine the iterative benefits of the development process, which are in large-scale operations necessary to consider for model updating, and what the triggers are for model retraining initiation.

The "Deploy Models" deployment pattern was the framework of choice as it matched the scope of the thesis project. It is the appropriate choice when model-training is non-iterative, or when the training load is too big and takes a long time. The deployment pattern is preferable when the development environments also have no strict separations. For example, in the use-case of the project, the dev- and staging environments were, in a sense, merged, as the evaluation and testing of the model occurred in the dev environment. The reason for merging the environments was data-oriented, as the model was automatically tested on the test set after model training. For the course of the project, the dev environment and the prod environment for both the model artifact and ancillary code were highly analogous to each other. This feature made updating the model in the prod environment faster after training a new model artifact in the dev environment. Databricks[BN22] does, however, recommend using the second pattern, "Deploy Code", for MLOps frameworks, instead of the "Deploy Models" pattern. The "Deploy Code" pattern promotes the code and the model towards the prod environment, where a new model is

trained in each environment. First in dev for model development, second in staging for integration testing, and finally in prod for the final model artifact that will be used in the application. The deployment pattern's environments are more strictly separated which makes development more structured. Both code and model performance are tested in staging, making the development process safer, and making it easier to automate retraining in locked-down environments, ensuring the reproducibility of ML models and ancillary code. The deployment pattern requires a CI/CD -infrastructure however that makes larger model artifacts, or models that are to be one-off developed, more cumbersome time-wise to train and deploy to production. The deployment pattern requires the adoption of modular code practices to be used efficiently, which demands more prior knowledge from team members unfamiliar with software engineering practices, such as Data Scientists. The "Deploy Code" deployment pattern is preferred over "Deploy Models" when a more strict structure and better scaling opportunities of the pipeline are required. However, for the use case in this project "Deploy Models" was favored, and similarities with the deployment pattern and the framework utilized for developing Oravizio showcases that it is applicable for certain development cases for clinical applications[GT21]. Further interest exists in determining what is a suitable deployment pattern, but that is beyond the scope of the project. It is important to consider that the development process and deployment patterns offered by Databricks Inc. are one of several alternatives for implementing MLOps practices and a more suitable approach may exist for clinical purposes. Furthermore, it is worth noting that MLOps-framework development is an ongoing process, and the best practices that are currently recommended based on the current knowledge and experience may evolve, slightly or significantly.

5.2 The MIMIC-IV Data

The MIMIC-IV data was the data set of choice for the project as it is a large set of real-life clinical data. It is imperative to utilize real-life numerical circumstances when training AI and ML models to enhance the feasibility and credibility of the resulting predictions for clinical application. The project team behind the resulting data consists of a team of interdisciplinary competencies, with data scientists, clinicians, and researchers who work to develop and maintain the data. This collaboration is crucial for interpreting the clinical context of the data. In the context of ML development, it highlights the importance of interdisciplinarity in defining the use case for the extracted data and the application of the ML model. The data is de-identified using a neural network with a transformer-architecture[JBP20] combined with a rule-based approach from a software package[Nea+08]. The deidentification process follows the stipulation formulated by the HIPAA Safe Harbor to remove 18 identifiers from a dataset to classify it as deidentified. The technique can be replicated when the deidentification of other electronic health records is necessary for approved regulatory use. However, the HIPAA is solely applicable in the United States. If an ML application is utilized in medical devices or other software applications in another regulatory market, for example, the EU, the data used to train the model must follow the requirements of the GDPR to be considered

de-identified. As HIPAA considers solely personal health information, and GDPR considers general personal information, the scope is larger for GDPR, the definition of the data to protect is different, and the acquisition requires different forms of consent[KD18]. this showcases the importance of responsibly processing the data acquisition to make the data and ML application compliant with current regulations in different areas. Databricks Inc. suggests that a Data governance officer should be involved in the MLOps team during the MLOps workflow, to ensure that data governance, data privacy, and other measures for compliance are adhered to across the data acquisition, model development, and deployment process. A vital consideration is the acquisition of data for MIMIC-IV. The data was collected from 2008 to 2019 from the ICU of Beth Israel Deaconess Medical Center(BIDMC) in Boston, Massachusetts, and is generated and archived as part of routine clinical care, monitoring, provider orders, and billing. To extract EHR data, it was collected from a MetaVision[IMD] clinical information system for bedside critical care[al23]. The data is first collected into a single data warehouse system and then transferred to secure servers for further processing. The process described was functional for MIMIC-IV, though a similar process to collect real-life clinical data from another hospital may need to differ considerably. If data acquisition is to occur at multiple hospitals, a highly adaptable acquisition method is necessary for minimal disruptions.

5.3 MLflow

MLflow proved to be a necessary component for enabling MLOps practices throughout the project, in particular the model monitoring. It was the software library of choice for model management as it was recommended, and developed, by Databricks Inc.[BN22] It was therefore a natural choice as the MLOps practices utilized were based on the content from the white book.

5.3.1 Usage of the MLflow package

MLflow usage was based on the MLOps development process and -development pattern showcased respectively in figures 3.1 and 3.2. As data was collected, analyzed, and feature-engineered based on earlier research and available resources, MLflow was mainly used for model training, validation, deployment, and monitoring. For model training, The MLflow Tracking 3.3.2 component, together with the MLflow Model Registry 3.3.2, were used to track artifacts produced from model training showcased in the MLflow UI. After testing the trained models' performance, the evaluation metric artifacts were logged with the corresponding model artifact. Model validation occurred using the MLflow UI, where the metric performance for each model was visually compared, and the model with the highest overall metrics performance was selected for model deployment and -monitoring. Model deployment used the 3.3.2 component to serve the model artifact to a local host for model inference with data ingested using the MLflow Tracking REST API. The MLflow Projects 3.3.2 component was unused during the project. The component is designed for reproducibility of data science code, either for code-sharing or for automation purposes such

as unit testing or automated runs for model retraining. This was, however, in some aspects both irrelevant and beyond the scope of the project as the software was managed in a local environment with the required software library dependencies readily available, and to scale the project for automation, an iterative development process or further model utilization was considered beyond the objective for the project. The MLflow Projects component can be considered a necessary component for automation and simplified code distribution.

5.3.2 Interpretation of utilization

An important consideration is that the MLflow implementation used was curated for this certain use case, and will differ when fully applied to a clinical setting. In the case of ventilator devices in the ICU, a full implementation of the MLOps framework would have to consider how to curate and deidentify patient data, how the distribution of model artifacts should occur, and how re-training would consider data locality, as demonstrated with the altered MIMIC-IV data. Furthermore, The MLflow library has developed considerably throughout the project. At the initial implementation of MLflow, the library was at version 2.2.2. Now the library is in version 2.9.2, where the amount of concepts has increased with less rigid definitions, such that MLflow is no longer defined primarily by the components described in section 3.3.2. This does not deprecate the utilization of the MLflow library throughout the project as the features and functions detailed in section 3.3.2 are still highly relevant. Rather, they are no longer the main features of the software library[23b], as it has been expanded upon throughout the year 2023 in conjunction with the commercialization of the Large Language Model(LLM) architecture[Cav23]. These additions to the software library would have also contributed to the MLOps framework in the project, though implementing these additions would go beyond the limitations set in section 1.2.

5.4 Data Alteration

The alteration of the MIMIC-IV data set and the resulting exclusion of features are significant in the performance aspect of the TPC model and the implications of the features selected for the model to interpret. The motivation for the split was to cause an alteration of the data that would result in a performance drift, which it did in this case, and to illustrate how a minor alteration such as age difference can cause a major loss of performance. The model performance will be further discussed in section 5.6.
The exclusion of features from both sets of the altered data is significant as it might lead to variation and alteration in model performance. The motivation for feature exclusion is understandable as features with low occurrence should not be considered in predictions as it can cause overfitting and reduce model accuracy. The issue that occurs with the altered data is that exclusion for a feature in one data set does not hold validity for the exclusion of the same

feature in the other. For the resulting training data set showcased in table 4.1, 5020 of the patients, approximately 13.6% of the patients, had Medicaid insurance. However, the prevalence of Medicaid insurance was a lot smaller, 507 patients, in the inference data set, which resulted in the exclusion of the feature in said data set. For the feature shapes to match, the Medicaid insurance feature was also excluded from the training data set, which resulted in these patients being labeled with no insurance. It is however difficult to determine the significance of the feature exclusion based on model prediction performance, as it is unknown whether the features excluded were significant in the original model usage. This causes a "Black Box"-problem, where it is uncertain how the decision-making of the model functions. The original paper uses integrated gradients to determine feature importance for the eICU data set[RLH21]. This clarifies the decision-making of the model as the features and values of the eICU and MIMIC-IV data sets were feature-engineered to be similar. They are however still two different data sets, and to make conclusions about the MIMIC-IV data set's feature's importance based on the results from the eICU data set is not valid. It would therefore be of interest to observe the SHAP-values[LL17] of the MIMIC-IV data set's features when the TPC model would use all available data, and what features are attributed to be the most important for the resulting predictions. The exclusion of features also brings into question the importance of feature selection. A Deep Learning model, such as the TPC model, should utilize select features based on their significance in affecting the resulting prediction. If there are features that can be excluded due to lack of prevalence or significance they should not be initially present. It is important to note the demographic of the patients in the MIMIC-IV data when considering the inclusion of the insurance feature for the TPC model. As most patients in the data set are citizens of the United States, the insurance matters as it determines the range of treatment a patient can get, and the quality of treatment. However in other countries, where insurance is not required for eligible healthcare, this feature would be considered a redundancy.

5.5 Model Training

It is important to distinguish how the resulting evaluation metrics in table 4.3 describe the model training performance. The MAD and the MSE are both metrics that naturally perform well the closer the predictions are to the mean and the labeled value respectively, as defined in section 3.4. However, both metrics are sensitive to large outliers and skewness in the data value. The LoS prediction is naturally positively skewed, as no negative values exist for these predictions. Therefore, while the metrics are valuable in determining performance, they are at risk of showcasing a faulty perception of the acquired results. The MAPE and MSLE are both metrics that are more robust to skewness. The MAPE is a percentage deviation relative to the predicted value. The MSLE utilizes the log function to reduce large outliers' effect on the resulting error margin. A consideration for the MAPE is that very small true values affect the error majorly, as a division with values closer to zero produces values closer to infinity. This error is addressed with the modification of the denomi-

nator value for the MAPE in section 3.4. The MSLE is utilized as an evaluation metric for similar arguments presented in section 2.5.1. Although it is used as the loss function for model training it is still a valid metric for evaluating the results. The R^2 is significant as it gives us an understanding of how the independent variables affect model variance. However, altering the number of independent variables can affect the performance of the metric, which compromises the resulting metric values. For a better understanding of how the independent variable amount affects the performance, it would be of interest to evaluate a multi-regression model, such as the TPC model, with the adjusted R^2 metric, as the number of independent variables is of accounted for. Another approach for validating usage of the R^2 metric is careful initial consideration of what features to include, and not alter feature quantity. conclusively, regarding the evaluation metrics in table 4.3, the MAPE and MSLE metrics are the performances to further consider when determining model performance. The learning-loss curve illustrated in 4.1 shows that the lowest for both the training- and validation loss are at epoch 10. However, a consideration is that the training loss keeps reducing to values below the validation curve. This is an indication of the potential overfitting of the data during model training. However, the validation-loss curve does not increase in value but decreases, indicating that overfitting may not have occurred. The disparity between the loss curves is still a matter of discussion. For further model training, it would be of consideration to include early stopping to avoid model training potentially resulting in an overfitted model.
The mortality prediction task of the TPC model was excluded from the project. Mortality prediction did benefit the performance of the model in the original work when utilized in conjunction with LoS prediction[RLH21], however, the ethical aspects of the prediction task are necessary to regard. Predicting the potential mortality of an ICU patient is a major task with ultimate outcomes, where faults in the model predictions potentially lead to disastrous consequences. It can harbor psychological strain on both patient and clinician[Pet+23]. Therefore, because of these difficulties that could occur in a clinical setting, the task was excluded.

5.6 Model Inference

It is important to note that the model serving solution used in this project would look different when utilized towards a medical device, such as a ventilator used in an ICU. For the project, inference predictions occurred utilizing the model-serving function from the MLflow Models component. The model artifact was served to a local HTTP port and was given data by utilizing the MLflow REST API. For a similar solution in a clinical setting, there would require scaling regarding the amount of software required, and consideration of safely connecting the ventilator to the framework.

From the results showcased in section 4.4, performance-monitoring of the TPC model was successful with the MLflow library functions, and it is clear there is a reduction in performance with the altered data being the culprit. In the case of the performance, it can be stated that underspecification has occurred when exposed to the altered data. The purpose of altering the data

was to determine whether there would be a significant drift in performance if demographic alterations to the data occurred. From a clinical perspective, demographic variety is anticipated, and if not considered during model development increases the risk of a demographic bias in resulting predictions[Che+23]. If eventual data drift occurs it is imperative to retrain the model for performance improvement. The iterative process of MLOps practices would benefit efficiency and safety while reducing effort. The project did not initiate model retraining as it would require more data for retraining. An alternative would be to add a subset of the inference data to the training data and evaluate the new inference performance on the remaining subset. This was however not considered due to time restrictions. Resultingly, the method detected poor performance, but no retraining was initiated. The retraining itself is a difficult task, however, if considered for a clinical setting. The acquisition of novel data has to regard the regulatory aspect of data governance and privacy for data sets containing patient information. An alternative for addressing the emerging data issues when facing model retraining would be implementing decentralized Federated Learning techniques while establishing the MLOps framework [BS23]. Data would not have to be exchanged or transferred, resulting in governance and privacy not being compromised. Furthermore, it would address issues with data disparity, such as demographic locality, where the re-trained model would be generalized and optimized.

6

Conclusion

Based on the scope of the thesis set in section 1.2 and 1.1, a proof-of-concept for model monitoring, utilizing an MLOps framework for clinical decision support, has been achieved. The results showcase a clear detection of performance drift during model monitoring, due to the altered demographic data, resulting in a major loss in performance of 41.13% and 44.64% for the MAPE and MSLE respectively. Based on the results and the discussion that followed, the appropriate metrics for monitoring are the MAPE and MSLE, and, with future consideration of feature quantity, the R^2 metric.
Limitations of the project are apparent, with the absence of model retraining, the need for evaluation metrics that showcase the important features, and the disparity between the project- and the potential clinical setting. Furthermore, it is debatable whether the MLOps practices utilized in the paper are appropriate for ML development for projects at a larger scale and for clinical purposes. There is, however, potential for implementation with the usage of other novel ML techniques with the systems that Getinge provides, showing promising development in further digitalization and ML support in clinical settings where CDSS is used, such as the ICU.

7

Ethical Aspects

7.1 Managing health record data

The MIMIC-IV data set used in the project is collected from routine clinical practice at Beth Israel Deaconess Medical Center (BIDMC) in the United States. As the data is based on private health records and data from telehealth systems, the user of the data must be certified by the website physionet.org. To be credentialed for usage of the MIMIC data, a user has to participate in an online course from the Collaborative Institutional Training Initiative program website (CITIprogram.org). The course, Data or Specimens Only Research, Covers the history of human subjects research for clinical studies and current laws and practices in the United States. The reason is not based on legal measurement, but that the user is knowledgeable and capable of proper care of sensitive data and good clinical research practice when handling said type of data. No consensual process has to be made due to the de-identified nature of the data.

Bibliography

[FC54] B. Farley and W. Clark. "Simulation of self-organizing systems by digital computer". In: *Transactions of the IRE Professional Group on Information Theory* 4.4 (1954), pp. 76–84. DOI: 10.1109/TIT.1954.1057468.

[EH77] Shortliffe EH. "MYCIN: A Knowledge-Based Computer Program Applied to Infectious Diseases." In: (1977), pp. 66–69. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2464549/.

[BS81] Gerry R Boss and J Edwin Seegmiller. "Age-related physiological changes and their clinical significance". In: *Western Journal of medicine* 135.6 (1981), p. 434.

[Sha86] G. L. Shaw. "Donald Hebb: The Organization of Behavior". In: *Brain Theory*. Ed. by Günther Palm and Ad Aertsen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 231–233. ISBN: 978-3-642-70911-1.

[LeC+89] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[G91] Baxt W. G. "Use of an artificial neural network for the diagnosis of myocardial infarction." In: *Annals of internal medicine* 115(11) (1991), pp. 843–848. DOI: https://doi.org/10.7326/0003-4819-115-11-843. URL: https://pubmed.ncbi.nlm.nih.gov/1952470/.

[Lo+95] Joseph Y. Lo et al. "Computer-aided diagnosis of breast cancer: Artificial neural network approach for optimized merging of mammographic features". In: *Academic Radiology* 2.10 (1995), pp. 841–850. ISSN: 1076-6332. DOI: https://doi.org/10.1016/S1076-6332(05)80057-1. URL: https://www.sciencedirect.com/science/article/pii/S1076633205800571.

[Nea+08] Ishna Neamatullah et al. "Automated de-identification of free-text medical records". In: *BMC medical informatics and decision making* 8.1 (2008), pp. 1–17.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[Abr+16] Michael David Abràmoff et al. “Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning”. In: *Investigative ophthalmology & visual science* 57.13 (2016), pp. 5200–5206.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Hea17] Jeff Heaton. “An Empirical Analysis of Feature Engineering for Predictive Modeling”. In: *CoRR* abs/1701.07852 (2017). arXiv: 1701.07852. URL: http://arxiv.org/abs/1701.07852.

[LL17] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[Som+17] SP Somashekhar et al. *Early experience with IBM Watson for Oncology (WFO) cognitive computing system for lung and colorectal cancer treatment.* 2017.

[Joh+18] Alistair E W Johnson et al. *The MIMIC Code Repository: enabling reproducibility in critical care research*. https://github.com/MIT-LCP/mimic-code. 2018.

[Kay18] Mehmet Kayaalp. “Patient privacy in the era of big data”. In: *Balkan medical journal* 35.1 (2018), pp. 8–17. URL: https://pubmed.ncbi.nlm.nih.gov/28903886/.

[KD18] Hemant Pathak Nathan Leong Jackie Haydock Melanie Scott-Bennett Lisa J Acevedo Kathleen D Kenney and Lindsay R Dailey. “GDPR Implementation and HIPAA Compliance: An Analysis of the GDPR and HIPAA for U.S. Health and Life Sciences Organizations”. In: (2018). URL: https://download.microsoft.com/download/B/B/F/BBFc0412-E610-49D9-AF83-D76DE35259F7/GDPR_Implementation_and_HIPAA_Compliance_EN_US.pdf.

[Mat18] Aaron Davidson et al. Matei Zaharia Andrew Chen. *Accelerating the Machine Learning Lifecycle with MLflow*. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. (Accessed on 10/09/2023). 2018.

[MM18] Rob van der Meulen and Thomas. Gartner McGall. “Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence”. In: (2018). URL: https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence.

[Luc19] Yves-Alexander de Montjoye Luc Rocher Julien M Hendrickx. *Estimating the success of re-identifications in incomplete datasets using generative models — Nature Communications*. https://www.nature.com/articles/s41467-019-10933-3. (Accessed on 07/14/2023). 2019.

[N+19] Tomašev N et al. *A Clinically Applicable Approach to Continuous Prediction of Future Acute Kidney Injury - PMC*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6722431/. (Accessed on 05/17/2023). 2019.

[Sch+19] Christopher G Schwarz et al. "Identification of anonymous MRI research participants with face-recognition software". In: *New England Journal of Medicine* 381.17 (2019), pp. 1684–1686.

[DAm+20] Alexander D'Amour et al. *Underspecification Presents Challenges for Credibility in Modern Machine Learning*. 2020. arXiv: 2011.03395 [cs.LG]. URL: https://arxiv.org/abs/2011.03395.

[Hea20] Will Douglas Heaven. *The way we train AI is fundamentally flawed — MIT Technology Review*. https://www.technologyreview.com/2020/11/18/1012234/training-machine-learning-broken-real-world-heath-nlp-computer-vision/. (Accessed on 05/17/2023). 2020.

[JBP20] Alistair EW Johnson, Lucas Bulgarelli, and Tom J Pollard. "Deidentification of free-text medical records using pre-trained bidirectional transformers". In: *Proceedings of the ACM Conference on Health, Inference, and Learning*. 2020, pp. 214–221.

[Sut+20] Reed T Sutton et al. "An overview of clinical decision support systems: benefits, risks, and strategies for success". In: *NPJ digital medicine* 3.1 (2020), p. 17.

[Wyn+20] Laure Wynants et al. "Prediction models for diagnosis and prognosis of covid-19: systematic review and critical appraisal". In: *bmj* 369 (2020).

[FA21a] Food and Drugs Administration. *Artificial Intelligence and Machine Learning in Software as a Medical Device*. 2021. URL: https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-software-medical-device#regulation.

[FA21b] Food and Drugs Administration. *Good Machine Learning Practice for Medical Device Development: Guiding Principles*. 2021. URL: https://www.fda.gov/medical-devices/software-medical-device-samd/good-machine-learning-practice-medical-device-development-guiding-principles.

[Fra21] Klein MJ Frana PL. *Encyclopedia of Artificial Intelligence: the Past, Present, and Future of AI*. 2021. URL: https://ebookcentral.proquest.com/lib/lund/detail.action?docID=6526148.

[GT21] Stirbu. V Granlund. T and Mikkonen. T. "Towards Regulatory-Compliant MLOps: Oravizio's Journey from a Machine Learning Experiment to a Deployed Certified Medical Product". In: *Computer Science Springer Nature* (2021). URL: https://link.springer.com/article/10.1007/s42979-021-00726-1.

[Res21] Grand View Research. *Artificial Intelligence In Healthcare Market Size, Share, And Trends Analysis Report By Component (Software Solutions, Hardware, Services), By Application (Virtual Assistants, Connected Machines), By Region, And Segment Forecasts, 2023 - 2030*. web page. 2021. URL: https://www.grandviewresearch.com/industry-analysis/artificial-intelligence-ai-healthcare-market/methodology(accessed%20February%2028,%202023).

[Roc21] Emma Rocheteau. *Patient Outcome Prediction with TPC Networks*. https://github.com/EmmaRocheteau/TPC-LoS-prediction. 2021.

[RLH21] Emma Rocheteau, Pietro Liò, and Stephanie Hyland. "Temporal Pointwise Convolutional Networks for Length of Stay Prediction in the Intensive Care Unit". In: *Proceedings of the Conference on Health, Inference, and Learning*. CHIL '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 58–68. ISBN: 9781450383592. DOI: 10.1145/3450439.3451860. URL: https://doi.org/10.1145/3450439.3451860.

[BN22] Thomson. M Bradley. J Kurlansik. R and Turbitt. N. *The Big Book of MLOps*. 2022. URL: https://www.databricks.com/resources/ebook/the-big-book-of-mlops?itm_data=glossary-mlops-banner.

[GK22] Snehal Gadge and Aarti Karande. "Study of Different Types of Evaluation Methods in Classification and Regression". In: *2022 IEEE Region 10 Symposium (TENSYMP)*. 2022, pp. 1–5. DOI: 10.1109/TENSYMP54529.2022.9864426.

[HH22] Sarah L. Harris and David Harris. "3 - Sequential Logic Design". In: *Digital Design and Computer Architecture*. Ed. by Sarah L. Harris and David Harris. Morgan Kaufmann, 2022, p. 129. ISBN: 978-0-12-820064-3. DOI: https://doi.org/10.1016/B978-0-12-820064-3.00003-9. URL: https://www.sciencedirect.com/science/article/pii/B9780128200643000039.

[Jon22] Jeffrey M. Jones. "More in U.S. Retiring, or Planning to Retire, Later". In: (2022). URL: https://news.gallup.com/poll/394943/retiring-planning-retire-later.aspx.

[Jun22] Alexander Jung. "Components of ML". In: *Machine Learning: The Basics*. Singapore: Springer Nature Singapore, 2022, pp. 19–56. ISBN: 978-981-16-8193-6. DOI: 10.1007/978-981-16-8193-6_2. URL: https://doi.org/10.1007/978-981-16-8193-6_2.

[ME22] Cox T. Neale M. de la Marck K. Hellström I. Baku A. and Peter E. *The MLOps Road Map 2022*. 2022. URL: https://github.com/cdfoundation/sig-mlops/blob/main/roadmap/2022/MLOpsRoadmap2022.md.

[McC+22] Richard V McCarthy et al. "Applying predictive analytics". In: Springer, 2022.

[Mir22] Shahbazian R Mirtaheri S.L. "Machine Learning". In: *Machine Learning: Theory to Applications (1st ed.)* CRC Press, 2022, pp. 18–19. DOI: 10.1201/9781003119258. URL: https://doi-org.ludwig.lub.lu.se/10.1201/9781003119258.

[Sin22] Sinha G.R. Singh B.K. "Introduction to Machine Learning". In: *Machine Learning in Healthcare: Fundamentals and Recent Applications (1st ed.)*. CRC Press, 2022, pp. 107–109. DOI: 10.1201/9781003097808. URL: https://doi-org.ludwig.lub.lu.se/10.1201/9781003097808.

[Sta22] Statista. "Breakdown of software development methodologies practiced worldwide in 2022". In: (2022). URL: https://www.statista.com/statistics/1233917/software-development-methodologies-practiced/#statisticContainer.

[al23] Johnson AEW et al. "MIMIC-IV, a freely accessible electronic health record dataset". In: (2023). DOI: https://doi.org/10.1038/s41597-022-01899-x(AccessedFebruary7,2023).

[BS23] Laveen Bhatia and Saeed Samet. "A decentralized data evaluation framework in federated learning". In: *Blockchain: Research and Applications* 4.4 (2023), p. 100152. URL: https://www.sciencedirect.com/science/article/pii/S2096720923000271.

[Cav23] Fabio Caversan. "Making Sense Of The Chatter: The Rapid Growth Of Large Language Models". In: (2023). URL: https://www.forbes.com/sites/forbestechcouncil/2023/06/20/making-sense-of-the-chatter-the-rapid-growth-of-large-language-models/?sh=6a1222f56b33.

[Che+23] Richard J Chen et al. "Algorithmic fairness in artificial intelligence for medicine and healthcare". In: *Nature biomedical engineering* 7.6 (2023), pp. 719–742.

[23a] *Machine Learning Operations*. 2023. URL: https://ml-ops.org/.

[23b] *MLflow 2.9.2 New Features*. 2023. URL: https://mlflow.org/docs/2.9.2/new-features/index.html.

[23c] *MLflow Model Registry*. 2023. URL: https://mlflow.org/docs/latest/model-registry.html.

[23d] *MLflow Models*. 2023. URL: https://mlflow.org/docs/latest/models.html.

[23e] *MLflow Projects*. 2023. URL: https://mlflow.org/docs/latest/projects.html.

[23f] *MLflow Tracking*. 2023. URL: https://mlflow.org/docs/latest/tracking.html.

[Pet+23] Lena Petersson et al. "Ethical considerations in implementing AI for mortality prediction in the emergency department: Linking theory and practice". In: *Digital Health* 9 (2023).

[Tea23] The Google Cloud Team. *MLOps: Continuous delivery and automation pipelines in machine learning*. 2023. URL: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning.

[IMD] IMDsoft. *MetaVision ICU*. URL: https://www.imd-soft.com/metavision-products/mv-for-intensive-care-adults-neonatal-and-pediatric.