KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

GROUP T LEUVEN CAMPUS

Early prediction of sepsis using clinical data

Jeroen Vastmans

Rui Zhu

Supervisor(s): Prof. Bart Vanrumste

Co-supervisor(s): Chetanya Puri

Master's Thesis submitted to obtain the degree of Master of Science in Electronics and ICT Engineering Technology

Academic year 2020-2021

Acknowledgements

After months of hard work on this master's thesis, I was able to complete it successfully. However, this would not have been possible without the help and support of several people.

First and foremost, I would like to thank my supervisor Professor Vanrumste for allowing me to do this thesis and work on this interesting topic. Special thanks also go to my daily supervisor Chetanya Puri for his guidance during this thesis, his availability during the corona pandemic where everything had to be done remotely. I would also like to thank him for his feedback, guidance and patience during this thesis.

Secondly, I would also like to acknowledge Rui Zhu, a master student from Group T. He was my partner for this thesis, and I want to thank him for working with me on this thesis. I would also like to thank Rui for his hospitality when I came to work with him in Leuven at his place for the thesis.

Thirdly, I would also like to thank my brother Pieter Vastmans for helping me with any problems I had while writing and for proofreading this thesis. His critical view of the thesis provided us with good constructive feedback on the thesis.

Finally, I would like to thank both my parents for their encouragement throughout my studies and for the tips they gave me during the writing of my thesis. I could not have achieved all this without their continuous support.

Much gratitude to each of you.

Jeroen Vastmans

Foremost, I would like to express my deep and sincere gratitude to my research promotor, Professor Vanrumste, for giving me the opportunity to do research. I am also extremely grateful for my daily supervisor Chetanya Puri. Thank him for his guidance and patience through the process of my master thesis.

Secondly, I would like to thank my teammate, Jeroen Vastmans. I want to thank him for his hard work and contribution. Thanks for the patience, understanding and support. Without him, I could not have finished the thesis.

Finally, I would like to thank my parents for their support and encouragement alongside my studies.

Thanks to all the people who have supported me to complete the research work.

Rui Zhu

Abstract (NL)

Patiënten die worden opgenomen op de intensieve zorgen (IZ) hebben vaak ernstige en acute aandoeningen. Sepsis is een vaak voorkomende aandoening bij patiënten op de intensieve zorgen. Sepsis wordt veroorzaakt door de reactie van het menselijke immuunsysteem op een infectie, welke grote schade kan veroorzaken aan het lichaam. Wanneer deze infectie het lichaam binnendringt, zal het lichaam verzwakken en zal dit leiden tot het falen van de vitale organen of zelfs tot overlijden. Om dit te voorkomen, is er nood aan een systeem om het optreden van sepsis vroegtijdig en betrouwbaar te kunnen voorspellen.

In deze thesis werd een algoritme ontwikkeld om sepsis betrouwbaar te kunnen voorspellen gebruikmakend van voorziene patiëntengegevens. Voor deze thesis werden klinische patiëntengegevens van de PhysioNet Challenge 2019 gebruikt. Dit is een wedstrijd die uitgeschreven werd voor het ontwikkelen van geautomatiseerde algoritmes om het vroegtijdig optreden van sepsis betrouwbaar te kunnen detecteren op basis van voorziene patiëntgegevens. Echter bevatte deze data veel ontbrekende medische gegevens, hierdoor kon er niet direct een voorspelling gedaan worden naar het optreden van sepsis. We onderscheiden hoofzakelijk twee problemen: hoe de ontbrekende gegevens betrouwbaar ingevuld kunnen worden en hoe een vroegtijdige en betrouwbare sepsis voorspelling gedaan kon worden met de "ingevulde" gegevens. Dit werd verder onderzocht in deze thesis. Hiervoor werd er onderzoek gedaan naar de bestaande methodes die gebruikt konden worden voor het "invullen" van de ontbrekende gegevens en naar bestaande implementaties voor het betrouwbaar detecteren van sepsis. De geïmplementeerde algoritmes werden geëvalueerd met de ontworpen utiliteitsscore van de PhysioNet Challenge.

Voor het opvullen van de ontbrekende gegevens werden er experimenten uitgevoerd om deze te bepalen met de gemiddelde waarden van K naburige buren (KNN), het gemiddelde van de gegevens, vullen met nul en het voorwaartse vullen van de ontbrekende medische gegevens met gekende data. Nog resterende ontbrekende gegevens werden bepaald met het gemiddelde van de medische data of met nul. Uit deze experimenten bleek dat het vullen met nul en met het gemiddelde van de medische data, de beste performantie scores gaven met een utiliteitsscore van 0.087 en 0.086. De performantie van het ontwikkelde algoritme kan verhoogd worden door het toevoegen van een sliding window, de SIRS en SOFA medische sepsis evaluatie-scores en door het weglaten van kolommen die veel ontbrekende gegevens bevatten. Om erachter te komen welke feature optimalisaties voor de hoogste performantie toename zorgden, werden er experimenten uitgevoerd voor deze extra features. Hieruit kon afgeleid worden dat de SIRS en SOFA-scores een negatieve invloed hadden op de performantie van het algoritme.

De conclusie van deze thesis is dat het algoritme welke gebruik maakt van volgende configuratie: een beslissingsboom classificator, voorwaarts opvullen van de ontbrekende gegevens gevolgd door vullen met nul, een sliding window van zes uur voor de vitale functies van de patiënt en het weglaten van de kolommen waar meer dan 92% van de gegevens ontbreken, een gemiddelde utiliteitsscore van 0.127 behaalde in een 5-Fold cross validatie.

Key words: ICU, Machine learning, PhysioNet Challenge 2019, Sepsis, Vroegtijdige detectie van sepsis.

Abstract (Eng.)

Patients who are admitted to the intensive care unit (ICU) often have serious and acute conditions. Sepsis is a common acute condition among intensive care patients. Sepsis is caused by the human immune system reacting intensely to an infection, which can cause great damage to the human body. If this infection enters the body, it weakens the body and could lead to failure of vital organs or even death. To prevent this, there is a need for a system that predicts the occurrence of sepsis early and reliably.

In this thesis, an algorithm was developed to predict sepsis reliably using provided patient data. For this thesis, clinical patient data from the PhysioNet Challenge 2019 was used. This is a competition for the development of automated algorithms to reliably detect the early onset of sepsis using provided patient data. However, this data contained a lot of missing medical data, which made it impossible to directly predict the occurrence of sepsis. We mainly distinguish two problems: how to fill in the missing data reliably and how to make an early and reliable sepsis prediction, were examined in this thesis. This was further explored in this thesis. For this purpose, the various existing methods that could be used to fill the missing data and existing implementations for the reliable detection of sepsis were investigated. The implemented algorithms were evaluated with the designed utility score for the PhysioNet challenge.

To impute the missing data, experiments were performed to fill them with the mean values of k-nearest neighbours (KNN), the mean of the data, filling with zero and forward filling of the missing medical data with known data. The remaining missing data were filled with the mean of the medical data or with zero. These experiments showed that filling with zero and filling with the mean of the medical data gave the best performance scores with utility scores of 0.087 and 0.086. The performance of the developed algorithm was increased by adding a sliding window, adding the SIRS and SOFA medical sepsis evaluation scores and dropping columns containing many missing data. To find out which feature optimization gives the highest performance gain, experiments were conducted with these additional features. It could be concluded that the SIRS and SOFA scores had a negative influence on the performance of the prediction algorithm.

This thesis concludes that an algorithm that uses the following configuration: a Decision Tree classifier, forward filling plus zero filling of the missing data, a six-hour sliding window for the patient's vital signs, and dropping columns with more than 92% missing data, achieved, with an average utility score of 0.127, in a 5-fold cross-validation.

Keywords: Early detection of Sepsis, ICU, Machine learning, PhysioNet Challenge 2019, Sepsis.

Extended abstract

Sepsis is a life-threatening syndrome caused by a dysregulated immune response of the body to an infection. It is a major health issue with high morbidity and mortality. Every year around 11 million people die from it globally and $62 billion is spent in the US alone. Twenty per cent of the in-hospital deaths are caused by sepsis [1]. Early detection and treatment are of vital importance in the battle against sepsis. Research has shown that for every hour of delay in the administration of antibiotics, there is an associated 6% rise in mortality [2], [3]. However, it is very hard to detect sepsis in its early stage. Delay is always generated before diagnosis, while sepsis can deteriorate very fast. When the typical symptoms of sepsis manifest, it has already progressed to a certain degree, and the patient's life is already in danger.

In this research, we developed a machine-learning algorithm to predict the occurrence of sepsis for each hour with clinical data. The dataset was provided by the PhysioNet Challenge 2019. It contained clinical data of 40336 patients. For each patient, the data recorded the 41 attributes for about 40 to 50 hours. The 41 attributes consisted of eight vital signs, 26 laboratory measurement values, six demographics and one sepsis label as the outcome. The dataset contained a large quantity of missing value. Especially for the laboratory measurement values, most of them had more than 90% of data missing. Therefore, imputing the missing data was unavoidable before designing the algorithm. How can the missing data be imputed reliably, and which algorithm can predict the occurrence of sepsis accurately were the two research questions in this thesis.

To evaluate the performance of different data imputation methods and algorithms, a united evaluation metric was needed. In this paper, five evaluation metrics were applied, among which the utility score was the most important one. It took not only the accuracy but also the timeliness into consideration. Timely and accurate prediction for sepsis was awarded, while late or missed prediction of sepsis was penalized. That was why it can show the proficiency of the algorithm. In this paper, the K-Fold algorithm was applied for all the model training. Since each algorithm was performed K times on the dataset, the mean and stand derivation of the results were calculated to be compared. The mean of the utility scores was the most important evaluation metric in this paper.

During the experiment, the method or the procedure that gave the highest mean of the utility scores is chosen to be included in the algorithm. Those decreased or gave a low mean of the utility scores were discarded. In this way, after performing all the candidate methods and procedures, the combination of the methods and procedures that gave the highest mean of the utility score would come out and made up the algorithm we researched for. Firstly, different data filling methods and machine learning training models were applied to the dataset. The data filling methods included filling with the mean values of K neighbours (KNN), the mean of the data, zero and forward filling of the missing medical data followed by filling with the mean or zero.

Machine learning training models included Logistic Regression, Decision Tree, Random Forest, XGBoost, AdaBoost, Gradient Boosting and Light Gradient Boosting.

Results showed that filling with zero or the mean of the dataset gave the best performance. It was also showed the Decision Tree was the best training model. Combine the Decision Tree with the method of filling with zero or the mean of the dataset, achieved the means of utility scores of 0.087 and 0.086. However, it was low compared to the best results of the PhysioNet Challenge 2019.

Furthermore, we experimented by dropping columns containing too much missing data and adding the SIRS and SOFA medical sepsis evaluation scores and sliding window attributes. To find out which feature optimisation procedure can give higher performance, experiments were conducted for them one by one. From the experiments, it was concluded that both the SIRS and SOFA scores had a negative influence on the performance. Dropping columns containing more than 92% missing data and adding sliding window attributes were all beneficial to the performance. The forward filling plus zero filling method outperformed all the other data imputation methods after performing these procedures.

The final algorithm designed in this thesis used the following configuration: forward filling followed by zero filling, dropping columns with more than 92% missing data, a six-hour sliding window for the patient's vital signs, and a decision tree classifier. It achieved an average of 0.127 and a standard derivation of 0.026 for the utility score, the best performance for sepsis prediction on the full dataset in this thesis.

Keywords: Early detection of Sepsis, ICU, Machine learning, PhysioNet Challenge 2019, Sepsis.

List of abbreviations

ADA	AdaBoost
AI	Artificial Intelligence
AISE	Artificial Intelligence Sepsis Expert
AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the Receiver Operating Characteristic
BUN	Blood urea nitrogen
CUDA	Compute Unified Device Architecture
CV	Cross Validation
DT	Decision Tree
EGDT	Early Goal Directed Therapy
ESICM	European Society of Intensive Care Medicine
FN	False Negative
FPR	False Positive Rate
GRAD	Gradient Boosting
GRU-D	Gated Recurrent Unit with a Decay mechanism
HPC	High-Performance Computing
ICU	Intensive care unit
KNN	K-Nearest Neighbours
LCOF	Last Observation Carried Forward
LGBM/LightGBM	Light Gradient Boosting Machine
LODS	Logistic Organ Dysfunction System
LOS	Length of Stay
LR	Logistic Regression
MAP	Mean Arterial Pressure
ML	Machine Learning
MOOC	Massive Open Online Course
MRI	Magnetic Resonance Imaging
NaN	Not a Number

PR	Precision-Recall
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RR	Respiratory Rate
SCCM	Society of Critical Care Medicine
Sepsis-3	The Third International Consensus Definitions for Sepsis and Septic Shock
SIRS	Systemic Inflammatory Response Syndrome
SOFA	Sequential Organ Failure Assessment
SOTA	State Of The Art
STD	Standard Deviation
SVM	Support Vector Machines
TASP	Time phAsed mode for Sepsis Prediction
TP	True Positive
TPR	True Positive Rate
US	United States
WBC	White Blood Cells
XGB	XGBoost

List of Figures

List of Tables

TABLE OF CONTENTS

1 INTRODUCTION

Sepsis is a life-threatening syndrome caused by a dysregulated response of the body to an infection. Hard to detect, it can deteriorate rapidly and result in organ dysfunction or even death. Every year approximately 50 million people are affected by sepsis and around 11 million people die from it [4]. It has been the primary cause of hospital death. Besides high morbidity and mortality, sepsis imposes an immense cost on the healthcare system, exceeding those of any other health condition. Annually, approximate $62 billion is spent in the US alone [5].

The high fatality rate of sepsis always results from the delay before diagnosis and treatment. Researches have shown the significance of early identification in improving the outcome and reducing mortality [6]. Although criteria for recognizing sepsis have been proposed, it fails to identify sepsis early and accurately.

Artificial intelligence (AI) is a branch of computer science that generates multifaceted algorithms to solve complex real-life problems. With continuous development, AI has penetrated a myriad of applications in sciences and technology. Machine learning is a branch of AI that specifies data analysis and decision making. It has been applied to medical science, such as early diagnosis, outcome prediction, drug development and personalized prescription. Patients in the intensive care unit (ICU) are technologically dependent on round the clock monitoring and life-sustaining medical equipment. It creates an advantageous opportunity for collecting electronic medical measurement of a huge number of physiological parameters. This makes applying machine learning to sepsis prediction possible.

In this thesis, an algorithm implemented with a machine learning model is developed to predict the occurrence of sepsis from the clinical measurement recorded hourly for each patient. The data is offered by the PhysioNet Clinical Challenge 2019 - Early Prediction of Sepsis from Clinical Data [7], an international challenge organised among researchers around the world. It consists of the ICU patient's medical parameters as input and sepsis outcome as output for each hour. ICU data is prone to lead detachments, and other medical parameters such as laboratory values are missing in hourly administration. This leads to a sparsely populated dataset such that a large proportion of the data is missing. Handling of missing data is traditionally done via imputation or deletion of the missing rows. Imputation of the missing data is indispensable and performed before the model is trained for prediction.

A more detailed introduction to sepsis and the PhysioNet Clinical Challenge is presented next. Furthermore, we discuss the problem description and the research questions of the thesis. At the end of the introduction, an outline of all the chapters for this thesis is presented.

2 SITUATING THE MASTER'S THESIS

2.1 What is sepsis

Sepsis, also known as septicaemia, is the result of the reaction of the human immune system to an infection injuring the body's tissue and organs. When the infection attacks the body, organs will be damaged and will gradually lose their functions. In most cases, a bacterial infection is the cause of the occurrence of sepsis, but other organisms like viruses and fungi can also be the cause [8].

Common sepsis-related infections initiate from the bladder, lung, skin, brain, urinary tract, and abdominal organs [9]. When sepsis occurs, the patient's body reacts very strongly to the infection. This is the patient's immune system that tries to neutralize the "foreign" intruder. Once the bacteria are in the bloodstream, the patient's condition will deteriorate very quickly [10], [11].

Studies [12] have shown that it is important to seek immediate medical attention when a patient experiences multiple of the typical symptoms mentioned below. *The sooner the patient receives treatment, the higher the chances of survival*.

These typical symptoms of sepsis are:

- Rapid heartbeat.
- Fever or low body temperature (above 38 °C or below 36 °C)
- Rapid breathing or shortness of breath.
- Confusion

In Figure 2.1, the different stages of sepsis are shown. There are three different levels of sepsis: sepsis, severe sepsis and septic shock.

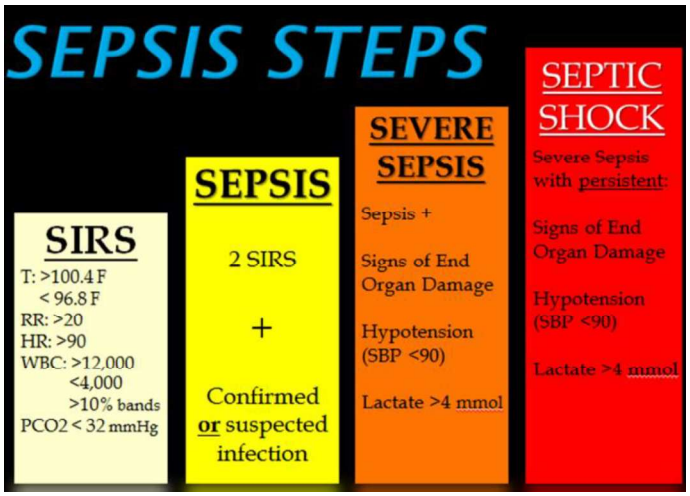

Figure 2.1: The stages of sepsis [55]

2.1.1 Sepsis

The first level of sepsis is simply called sepsis. The symptoms are also common for other "general" diseases. In addition to the general symptoms, an infection starts to occur in the patient's body. Patients are most likely diagnosed with sepsis when besides the typical sepsis symptoms there are signs of an infection or when the patient falls into one of the higher health risk groups (elderly people +65 years, with weakened immune systems or with a chronic medical condition like diabetes) [13].

In Figure 2.2, the common sepsis symptoms are represented. The acronym TIME is used for summing up the major symptoms.

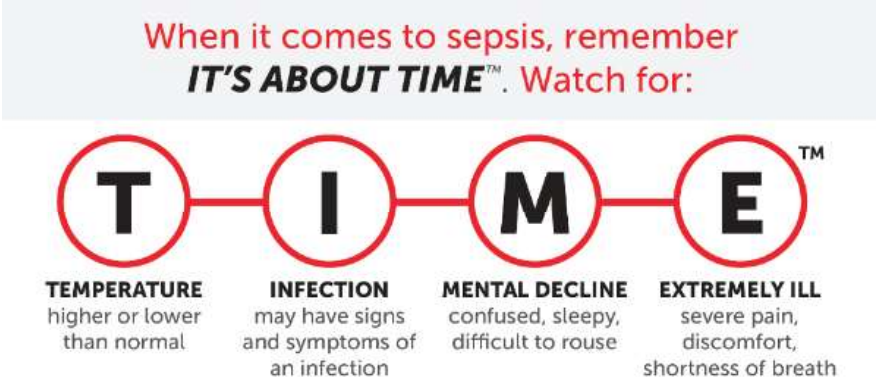

Figure 2.2: Common symptoms of sepsis [39]

2.1.2 Severe Sepsis

The second level of sepsis is called severe sepsis. Patients diagnosed with this level of sepsis have additional symptoms that are summed up below. These symptoms are always related to organ failure, which suggests sepsis has developed to a degree that interferes with the normal activity of the body organs [10]:

- Spots of discoloured skin
- Abnormal heart functions
- Breathing problems
- Decreased urination
- Low platelet (blood clotting cells) count
- Chills due to falling in body temperature
- Extreme weakness

2.1.3 Sepsis Shock

The third and most dangerous level of sepsis is septic shock. This is a very serious condition as a large number of bacteria enter the blood as a result of an infection and/or organ failure. These two conditions, infection and organ failure, combined with a deteriorated health condition will lead to a septic shock. In addition to the symptoms of the previous level, a septic shock will only occur when the patient has very low blood pressure. This blood pressure drop indicates that the patient's organs do not receive enough oxygen to function properly. Approximately half of the patients treated in intensive care units do not survive a septic shock [10].

2.2 PhysioNet Challenge

As mentioned in chapter 2.1, the early detection and treatment of sepsis may positively improve the patient's outcome when sepsis occurs. Although critical care societies have proposed new clinical criteria that can help with sepsis detection, the fundamental need for early detection and treatment remains unsolved. As a reaction to this problem, researchers have proposed different algorithms for early sepsis detection. Unfortunately, it is not possible to directly compare these algorithms, because they use different clinical variables, sepsis criteria, and patient associates as well as use different prediction methods and metrics to evaluate their performance. To address these issues, the challenge, PhysioNet Computing in Cardiology Challenge 2019 was organized [7]. This challenge has sped up the development of automated open-source algorithms for the early detection of sepsis using clinical data. For training purposes, a dataset was provided which consists of 40 hourly taken patient parameters (features). In chapter 4.1, this dataset will be discussed more in-depth. However, the dataset contains a lot of missing data gaps as shown in Figure 2.3.

Figure 2.3: Missing data visualisation. Each row represents hourly patient measurement present (black) or absent (white) for the features (columns)

The participants of this challenge were asked to implement working algorithms that could automatically predict the occurrence of sepsis in a patient based on the provided training data. Since the dataset contains missing data, the missing data needs to be imputed to reliably predict the onset of sepsis. Moreover, it was required that these algorithms can make a positive or negative sepsis prediction solely based on the hourly measurements. The challenge asks the participants to forecast the onset of sepsis at least six hours, and not more than twelve hours, before the occurrence of sepsis as indicated by the Sepsis-3 clinical standards [14].

The patient data is labelled with a positive Sepsis label ('1') for patients who develop sepsis at time $t = t_{sepsis} - 6$ and a negative Sepsis label ('0') for t < $t_{sepsis} - 6$, t_{sepsis} is the time of sepsis onset as defined by the Sepsis-3 definition [7], [15]. The data for patients who have never experienced sepsis is also labelled '0'. Predictions are evaluated based on their success in binary classification against a utility score function that is defined in the challenge summary. False negative predictions for non-septic patients are penalized and true negative predictions get a score of zero. Too much early prediction is penalized for septic patients, false negative (FN) predictions are more severely penalized and only true positive (TP) predictions result in a positive score.

The challenge participants' algorithms were evaluated using a new clinical utility score-based metric. This utility score, U(t), rewards algorithms for early sepsis predictions and penalizes them for late and missed sepsis predictions and also for false alarms (predict sepsis by a non-sepsis patient).

For this thesis, the provided data of the PhysioNet Challenge 2019 will be used to predict the early onset of sepsis.

2.3 Problem description

Every year, at least 11 million people die from the effects of sepsis. In other words, every 2.8 seconds someone in the world dies from sepsis. Globally, 20% of deaths are related to sepsis. In Europe, 700 out of 100,000 people are affected by sepsis every year [4]. The annual number of new cases of sepsis is higher than that of cancer. Sepsis can affect anyone, regardless of age, gender, or geography. However, some people are at higher risk: children, the elderly, the chronically ill (e.g., people with diabetes) and people with weakened immune systems (such as those who miss an organ or are treated with chemotherapy). Among patients who have survived sepsis, around 40% experience serious and persistent symptoms.

In 2019, new sepsis definitions were issued by the Society of Critical Care Medicine (SCCM) and the European Society of Intensive Care Medicine (ESICM) for screening and early identification. However, their benefits have yet to be validated by prospective studies [14], [16]–[18] and experts continue to emphasize the early administration of antibiotics and fluids for the initial resuscitation of patients with sepsis. Rapid detection of sepsis and starting the treatment early are still the most efficient way of combatting it.

Sepsis is life-threatening but treatable if diagnosed early. Early detection and treatment of sepsis using a sepsis protocol will result in a lower mortality rate. The early detection of the occurrence of sepsis is the key to help healthcare workers in starting up an efficient treatment for the patient more quickly.

In Figure 2.4 and Figure 2.5, a missing data graph and a graph that represents the evolution of the vital signs of a septic patient are plotted.

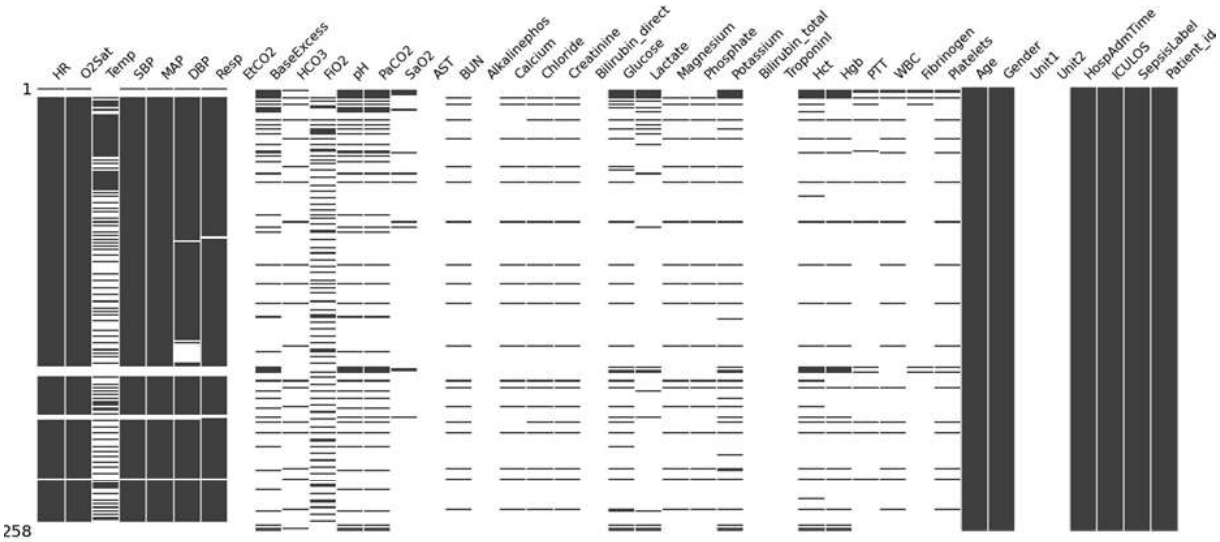

Figure 2.4: Visualisation of missing data of a septic patient

As shown in Figure 2.5, the hourly patient data contains numerous missing data gaps in the vital signs. In this case, the temperature variable of this patient has a lot of missing data. Since the data contains a lot of missing values, it cannot directly be used for predicting the onset of sepsis. Otherwise, this might lead to false sepsis predictions. Once the missing values are imputed, the onset of sepsis can be correctly and reliably predicted.

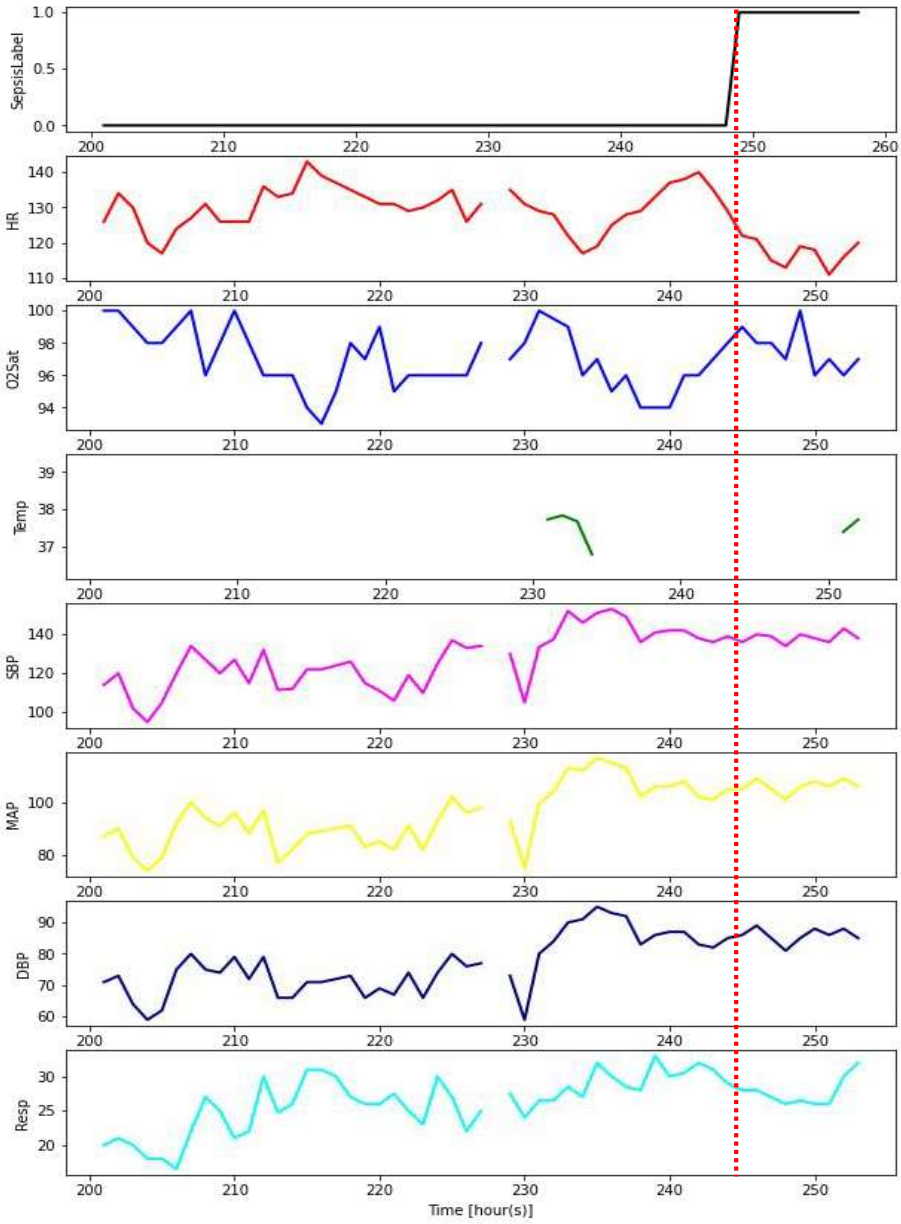

Figure 2.5: Evolution of the vital signs of a septic patient

2.4 Research questions

This thesis will deal with two research questions:

- How can the missing data be imputed reliably?
- Which algorithm can predict the occurrence of sepsis accurately?

We will investigate the possible methods for achieving the above-mentioned research questions. First, research will be conducted on the various existing methods for reliably imputing the missing data values in the provided data. Based on this research, experiments will be carried out to find the optimal (accuracy/time) imputation method which achieved the best performance. Next, research will be performed to investigate what the existing methods are for reliably determining the onset of sepsis at an early stage. For this purpose, research will be conducted on the already existing implementations for this problem. Since this master thesis is based on an open-source computing challenge, an analysis will be executed regarding the different implementations of participants and the results that were obtained by their designed algorithms. Based on this research, an algorithm will be created to reliably predict the sepsis onset. The research that will be conducted to answer these research questions will be discussed in the following chapters. We will also compare the performance/accuracy of our developed algorithm against the different implemented algorithms created by participants in the PhysioNet Challenge 2019.

3 LITERATURE STUDY

3.1 Discussion of Time series

This thesis uses pre-recorded medical data of ICU patients which is hourly collected. This provides the medical personnel with a better view of the medical evolution of a patient's condition. Since the data is taken hourly, we can say that this data is a time series. In the following chapter, we will briefly discuss time series and the application in this thesis.

3.1.1 Introduction to time series

A time series is a series of data points that are taken at even intervals in time and are indexed with time as the "indexing" parameter. Since the data points in the time series are collected at even times, there might be a correlation between the different data points [19]. In a time series, time is often the independent variable that can be used to forecast the future. The data which is used for this thesis consists of hourly measured medical vital signs and the corresponding output (Sepsis) label. Given that the output labels were provided, it is possible to predict the sepsis labels using a model that is trained on the medical features and the given sepsis labels.

In general, two types of time series models can be used for representing a dataset in time. These models are:

- Univariate time series models
- Multivariate time series models

3.1.2 The different time series models

3.1.2.1 *Univariate time series*

A univariate time series exists when a dataset contains only one time-dependent variable. When data of this time series is plotted, only the changes of the variable over time will be shown. An example of a univariate time series is the hourly measurement of the body temperature of a patient who has a fever, which is shown in Figure 3.1.

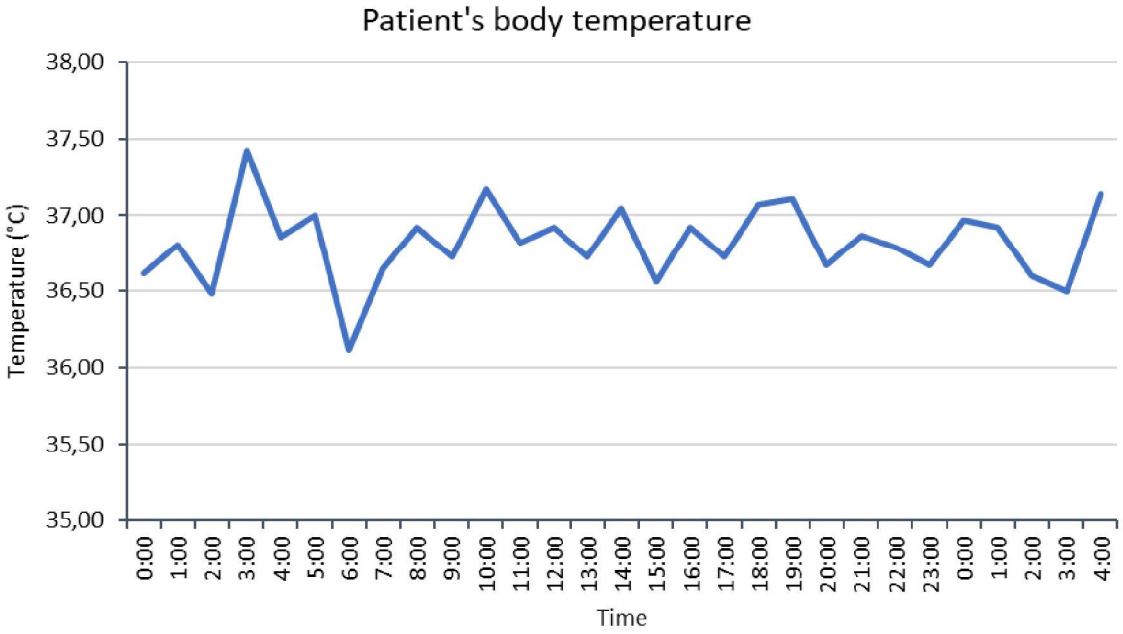

Figure 3.1: Example of univariate time series data

3.1.2.2 *Multivariate time series*

Unlike univariate time series data where there is only one time-dependent variable, multivariate time series data consists of multiple time-dependent variables. When analysing a multivariate time series, it may be assumed that the different time-dependent variables not only depend on the previous values of these variables but that there exists a correlation between these variables.

In Table 1, an example of multivariate time series data is given. The hourly measurements of different medical features regarding a patient's health and the corresponding output label are shown. The different medical features correlate with each other and the output label, based on these correlations a prediction of the output can be made. For example, there may be a correlation between a patient's age and other medical parameters and elderly people may have a higher heart rate and an increased chance of having sepsis. In Figure 3.2 and Figure 3.3, a subset of medical features of both a sepsis and non-sepsis patient is displayed.

Table 1: Example of a multivariate time series

Time	HR	O2Sat	Temp	EtCO2	Resp	Age	Patient_id	SepsisLabel
1	110	94	36.33	31.0	28	83	1	0
2	108	95	36.66	31.6	29	83	1	0
3	106	99	36.30	31.2	26	83	1	0
4	104	95	36.92	31.0	30	83	1	0
5	102	89	36.86	31.6	27	83	1	1
6	104	91	37.00	32.2	28	83	1	1
1	98	92	36.11	32.1	23	45	5	0
2	97	91	37.25	31.6	19	45	5	0
3	89	95	36.81	33.2	22	45	5	0
4	90	91	36.72	33.0	23	45	5	0
5	103	92	37.17	34.1	24	45	5	0

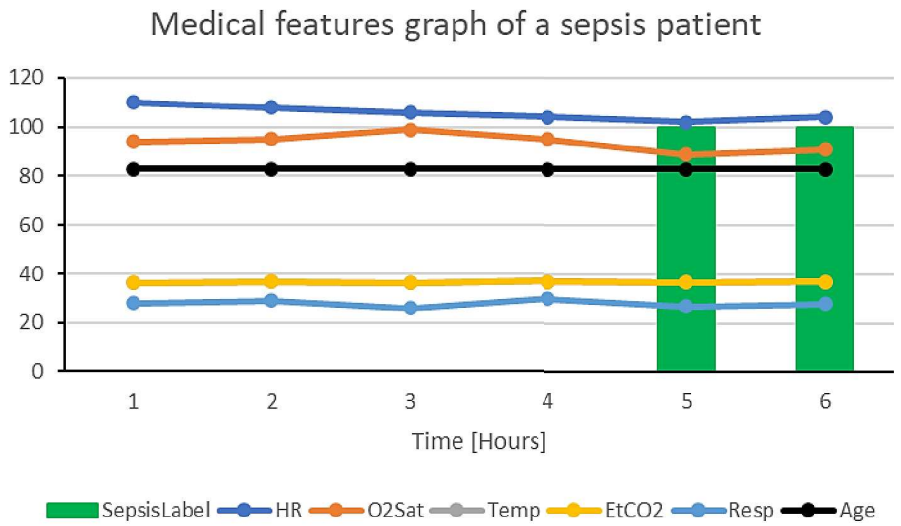

Figure 3.2: Graphical representation of the medical features of a sepsis patient

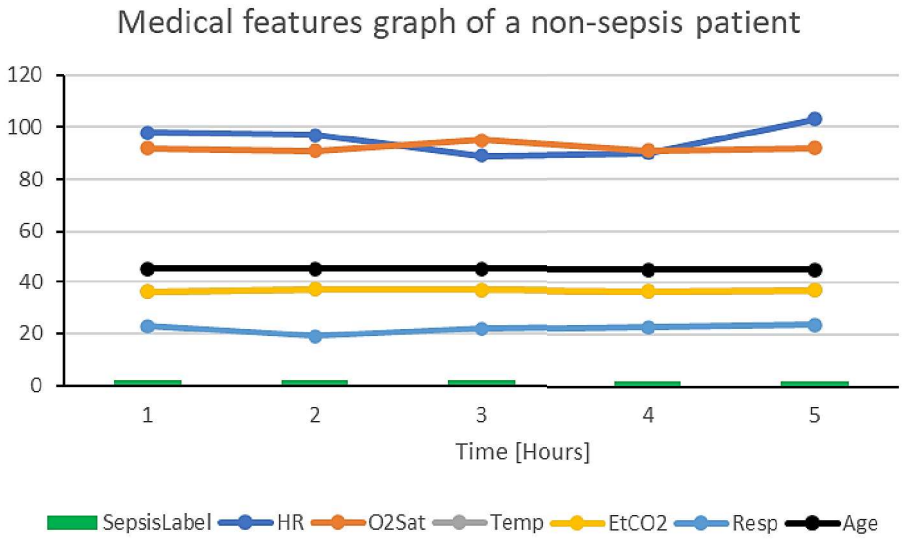

Figure 3.3: Graphical representation of the medical features of a non-sepsis patient

3.2 Related works

As mentioned in the introduction, this study focuses on the early detection of sepsis, for which the PhysioNet Challenge was organized. In Table 2, the five papers of the participants who had the highest utility score for the PhysioNet Challenge, are summarized. The PhysioNet competition proposed a new metric, called the "Utility Score" for evaluating time-wise prediction accuracy and models which correctly predicted the sepsis onset around 12 hours before a physician's diagnosis. The available training data consists of two parts, training data A and B that are obtained from two hospital systems. We discuss the data in-depth in chapter 4.1. The two last columns of Table 2 displays the mean utility score achieved by K-Fold cross-validation on the hidden full data and hidden dataset A & B for each of the submissions.

Table 2: Top 5 submissions for the PhysioNet Challenge

Publication	**Missing data imputation**	**Sepsis prediction approach**	**Utility Score on hidden test data**	**Utility Score on hidden dataset A & B**
The Signature-Based Model for Early Detection of Sepsis From Electronic Health Records in the Intensive Care Unit [20]	Forward filling method	Crafted features + signature Features Light Gradient Boosting - 5-fold Cross Validation (CV)	0.360	A: 0.433 B: 0.434
Automated Prediction of Sepsis Onset Using Gradient Boosted Decision Trees [21]	Log transformation + normalisation + mean filling	Creating new features + Gradient Tree boosting + 10-fold CV	0.345	A: 0.409 B: 0.396
Sepsis Prediction in Intensive Care Unit Using Ensemble of XGBoost Models[22]	Linear interpolation	Best 5 features + XGBoost + 5-fold CV	0.339	A: 0.422 B: 0.395
TASP A Time-Phased Model for Sepsis Prediction[23]	Forward filling missing data	TASP (Light GBM + RNN) + 10-fold CV	0.337	A: 0.420 B: 0.401
Utilizing Informative Missingness for Early Prediction of Sepsis [24]	No filling	Informative missingness (IM), sliding window + XGBoost + 5-fold CV	0.337	A: 0.401 B: 0.407

While the PhysioNet Challenge organizers posed a challenge of detecting sepsis in a timely manner, the provided data has missing data, making it difficult to train models to learn and predict sepsis. As a result, challenge participants used various strategies to deal with the missing data and train sepsis prediction models. The methods used by each paper to solve these two problems are mentioned in Table 2 as well as their utility score for the data. The utility score is an evaluation metric that was designed for the PhysioNet Challenge.

3.2.1 The Signature-Based Model for Early Detection of Sepsis from Electronic Health Records in the Intensive Care Unit

James Morrill et al. [20] achieved the highest overall utility score of all the challenge submissions. This research uses a signature-based machine learning method which generates the risk score for a patient to develop sepsis. The risk score is calculated using hourly averaged patient data from the time since ICU admission until the current time.

The forward filling method is used to impute the missing data values in the training dataset. When there is no previous existing feature value, the value is left as a Not a Number (NaN) value. Since a lot of medical data is missing, this research creates additional features. These additional features are:

- ShockIndex: the ratio of Heart Rate and Systolic Blood Pressure (SBP)
- BUN/CR: the ratio between Bilirubin and Creatinine
- PartialSOFA: Score of the SOFA components that are found in the challenge data
- SOFA Deterioration: Binary label which is equal to 1 if PartialSOFA is increased by 2 in the last 24 hours

Because these measurements were taken infrequently, the laboratory features and temperature vital sign are sparsely filled (high missing data score). This research assumes that the frequency at which hourly measurements are taken indicates the patient's health condition and severity. When measurements are more frequently taken, this means that doctors are more concerned with the patient's health condition. As a result, a new feature was included which counts the number of measurements taken during a given time window. Additionally, the minimum and maximum value of the eight vital signs using a sliding window were also included.

For the reliable and early prediction of sepsis, the stratified 5-fold cross-validation method is used. By using this method, each fold contains approximately the same number of time points and septic cases. The Light Gradient Boost Machine (LGBM) is used as the algorithm to regress against the predicted sepsis labels. Next, a gradient-free optimization algorithm is used to determine the optimal cut-off threshold for the sepsis probability for maximizing the utility score on the training dataset. Both the regressor and threshold are applied to the test set to make an overall prediction and evaluate the score. This model that is constructed with hand-crafted features and signatures achieves an average utility score of 0.430. This is the highest achieved score of the submissions.

The use of signature values significantly improves the utility score. By performing the signature transformation, important details from the time series are uncovered which can be used to classify cases of sepsis.

3.2.2 Automated Prediction of Sepsis Onset Using Gradient Boosted Decision Trees

In this research, two data pre-processing steps are performed. The first pre-processing step performs a log-transformation of the data. Following this data transformation, the second pre-processing step is executed where the data is normalized and the missing values in data are replaced by mean values until the first valid (not NaN) value. In Figure 3.4, the steps implemented by this paper are shown. After this operation, the next feature variables were replaced with the most recent present value. Following this missing data imputation step, new additional features are augmented. As shown in the feature block of Figure 3.4, three additional feature sets, besides the original 40 features, will be created.

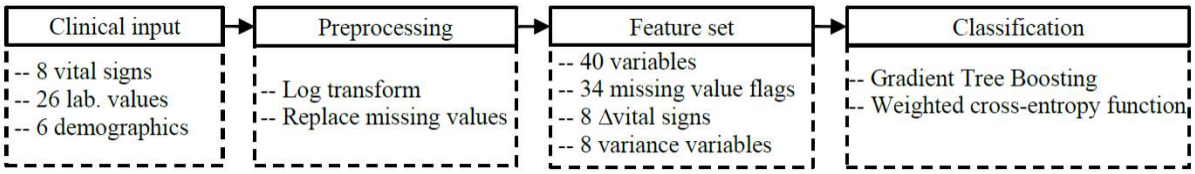

Figure 3.4: Data processing of paper 2 [21]

The first additional feature set that was used is a 34-length mask vector to indicate whether or not the feature data is missing. The mask has a value of one when the original feature was present and a value of zero when the original feature was missing and imputed at that time instance. The original feature set is supplemented with additional features like an 8-length delta, which tracks the changes of the vital sign variables at time steps, and an 8-length variance vector of the eight vital signs. As a result, depending on which additional feature sets are used, a vector with L=74/82/90 features is used for each hour of data. Before this vector is applied to the classifier, the features of the current hour are appended with the features of the previous 3 to 9 hours (H). But if the current hour is less than H, then the feature vector will be appended with zeros. Therefore, a vector with $\mathrm{L} * \mathrm{H}$ features is created and is fed into the classifier [21].

The algorithm's performance for each prediction is measured using the utility score defined by the challenge. The algorithm's performance for each prediction is measured using the utility score defined by the challenge. The utility function rewards classifiers who predict sepsis onset time 12 hours before and 3 hours after the onset time of sepsis.

A gradient Boosting model, with up to 30 trees, is used as a classifier to predict the early occurrence of sepsis. Various models were tested and compared using a stratified 3-fold. Models were trained for 40 epochs at a learning rate of 0.2. The submitted model was trained using 10-fold cross-validation, a learning rate of 0.1, and the AUPRC score of the validation dataset for each fold is monitored for early stopping [21].

From these experiments, they found that the best performance for a classifier was obtained when H=6, data of current hours and 6 previous hours, combined with the missing data mask and the variance vectors. This configuration obtained a mean utility score of 0.3981 on the holdout fold, after 3-fold validation.

However, they were not able to submit this model for testing in the official phase of the competition. But a model with H=5 combined with the base features and mask vectors was trained on the full training dataset and it obtained a 3-fold cross-validated score of 0.3933 and a 10-fold cross-validated score of 0.400. This trained model was submitted for final testing on the unseen test data, and it achieved the second-highest official score of the competition with a utility score of 0.345 on the complete test set.

3.2.3 Sepsis Prediction in Intensive Care Unit Using Ensemble of XGBoost Models

Compared to the other submissions of the top five, this paper makes use of ensemble model learning. The ensemble model learning method is a process where multiple models are created to predict an outcome. In this research, different XGBoost models were trained on the dataset, followed by the aggregation of the predictions of each model into an "ensembled" model.

In Figure 3.5, the steps for reliable prediction of sepsis implemented by this paper are shown. They implement two features pre-processing steps: feature extraction and feature selection. The "provided" dataset is divided into 2 parts. The split parts are respectively used for the feature selection step and 10% of the training data is used for the feature selection and tuning of the hyperparameters while the other 90% is used for training the ensemble classifier. A classifier is an algorithm that uses training data to understand how given input variables relate to output labels.

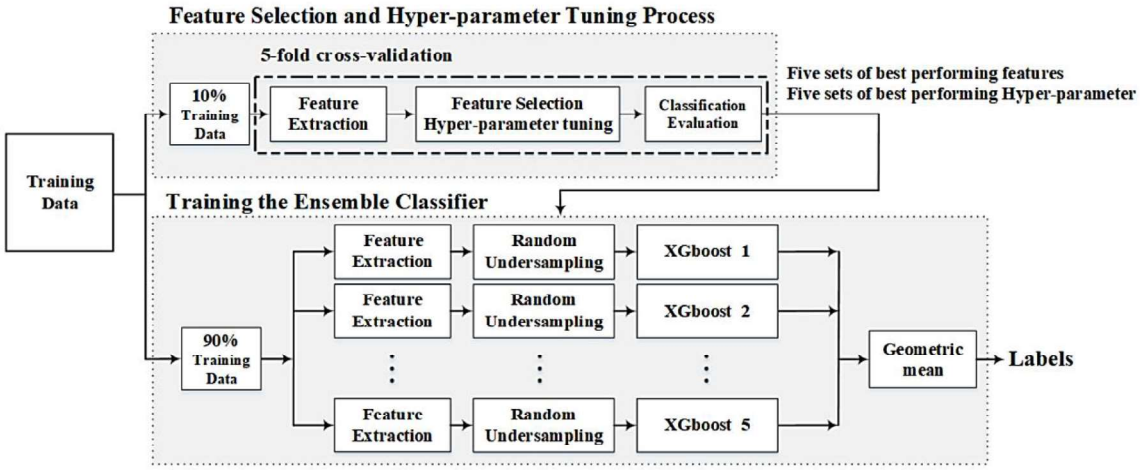

Figure 3.5: Training strategy of paper 3 [22]

In the feature extraction step, two different types of features are obtained from the dataset. The first type of feature identifies covariates that are missing less than 70% of the time. This set of features consists of the first 7 vital signs and the demographics features of the patient's data. Before this type of feature is extracted, the missing values are imputed using linear interpolation. But if all the values of the patient's observation are missing, then these values will be replaced by the mean value. When the missing data is imputed, the following feature sets are extracted [22]:

- Sliding window based features. For these types of features, the mean, minimum, maximum, median, variance, 95%, 99%, 5%, and 1% quantiles are calculated over two window sizes. Here the window size is the last 5 and 11 hours. Two different windows were used to gather the short and long-term evolution of the covariates.
- Non-sliding-window based features: The provided observations are used to measure energy, Shannon entropy, mean of the first differences, and observation lengths.
- The last observation values of the 13 covariates are used as "new" features.

The second type of features consists of all the features except the age and gender feature because they are constant during the patient's measurements. Sequence abstraction is used to represent the data absence. Each sequence is defined as a set of consecutive measurements where the values are either missing or present. This means that each sequence can only have missing or present values. For example, the HR measurement for 6 hours are {NaN, 95, 110, NaN, NaN, 100}. Based on the definition, the 4 sequences that can be abstracted are: {NaN}, {95,110}, {NaN, NaN}, {100}. Using the sequence abstraction, the following features are calculated [22]:

1. Mean and variance of the lengths of sequences along with each covariate, L_C
2. Summation and variance of the lengths of sequences with only valid values along with each covariate, L_{CV}
3. Mean and variance of the lengths of sequences along with each observation, L_O, in the last 5 hours

An example of the sequence abstraction is shown in Figure 3.6. It should be noted that the input clinical data varies in duration and the number of observations may be insufficient to exclude the functionalities of the sliding window. In these cases, the first observation is used to fill the clinical results. The padding sum corresponds to the difference between the number of observations in the data and the number of observations needed. This allows the raw data to be converted into a fixed-length function space. Conditional methods can also be used for such complex data, such as XGBoost and random forest [22].

Observation	HR	Temp	SBP	L_O
1	NaN	NAN	NaN	{3}
2	97	NaN	98	{1, 1, 1}
3	89	NaN	122	{1, 1, 1}
	90	NaN	NaN	{1, 2}
	103	NaN	122	{1, 1, 1}
.	110	NaN	NaN	{1, 2}
.	108	36.11	123	{3}
.	106	NaN	93	{1 1, 1}
	104	NaN	133	{1, 1, 1}
10	102	NaN	134	{1, 1, 1}
L_C	{1, 9}	{6, 1, 3}	{1, 2, 1, 1, 1, 4}	
L_{CV}	{9}	{1}	{2, 1, 4}	

Figure 3.6: Sequence abstraction for HR, Temp and SBP covariates [22]

The onset prediction of sepsis is done using a classification algorithm that consists of two steps. In the first step, five sets of the best performing features and hyper-parameters of each fold are selected. The feature selection and hyper-parameter tuning are performed using 5-fold cross-validation. The feature selection is performed using a wrapper feature selection algorithm which is based on XGBoost. The important metric is the number of times that a particular feature was split on in the XGBoost algorithm.

The tuning of the hyperparameters is implemented using the grid search method. This is a method that finds the (hyper) parameters that give the highest accuracy score for a given model, in this case, XGBoost.

In the second step, the different ensemble classifiers are trained. An ensemble of five XGBoost models is used for predicting the sepsis labels. An XGBoost model is a decision tree-based ensemble model that uses the gradient boosting paradigm. This ensemble model is trained using 90% of the (originally) provided dataset. However, five models need to be trained. Therefore, the 90% dataset is divided into 5 equal subsets. Each subset is used for training using one of the five distinct XGBoost models. The prediction model is created using the geometric means of the five model results. This model will be used to predict the (Sepsis) labels based on a given dataset.

The scores for the PhysioNet Challenge 2019 were achieved using the prediction model in a 5-fold cross-validation scheme using training data. Since the model is trained on the provided data (training dataset A and B), it has a high score for these unseen datasets. It received utility scores of 0.422 and 0.395 respectively for the unseen datasets A and B. However, the model performs wrong for the unseen dataset of a third hospital. It had a utility score of -0.146.

3.2.4 TASP A Time-Phased Model for Sepsis Prediction

Since the occurrence of sepsis is time-dependent, they developed a time-phased model for the early prediction of sepsis (TASP). In this research, a machine learning model is used to determine the risk for each hour of ICU stay. After a data analysis of the provided dataset, a relation was found between the occurrence of sepsis and the ICU Length of Stay (LOS) feature. A time-phased model was developed to predict the sepsis onset. In Figure 3.7, the composition of the TASP model is shown. There can be observed that different frameworks are used to predict the onset of sepsis for different durations of ICU LOS. The training data is sliced into three parts based on the length of stay in the ICU. These three parts [23] are:

- Early stage: This stage contains the data where ICU LOS is between 1 and 9 hours
- Middle Stage: Here, the ICU LOS feature is between 10 and 49 hours
- Late stage: ICU LOS is greater than 50 hours

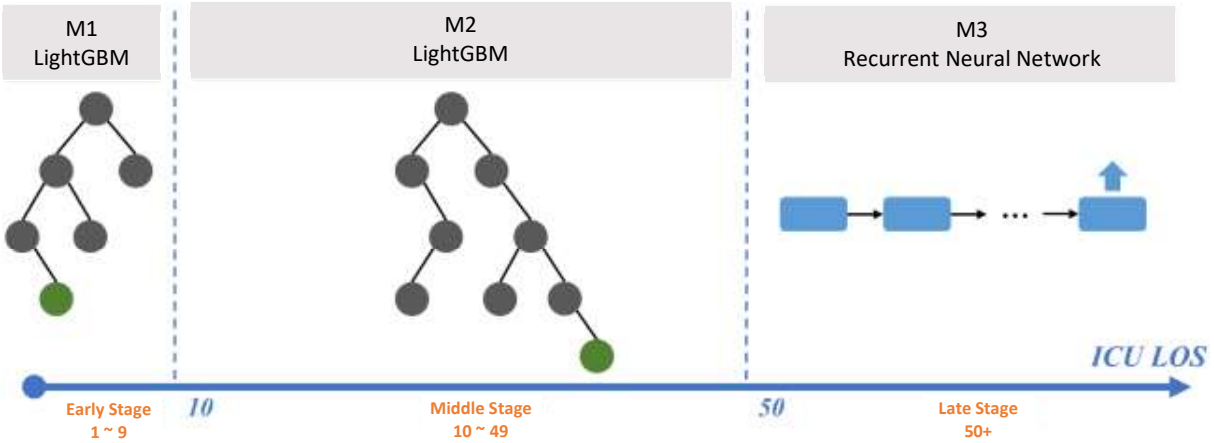

Figure 3.7: Architecture of the TASP model [23]

Data analysis is performed to examine the relation between the features and the missing data pattern. Three observations were made after data analysis.

The first observation is that the laboratory features have a high (> 90%) missing value score. When most of these values are missing (NaN value), it is difficult to impute these missing data parts for different learning methods. Next, there is an imbalance in the training data. Of the more than 40,000 patients, only 2932 patients (7,27%) develop sepsis during the ICU stay. The ratio of the sepsis labels is 98,2% for the negative label (Sepsis Label = 0) and 1,8% for the positive label (Sepsis label = 1). The final observation is that the incidence (number of new sepsis diagnosis/hour) of sepsis is non-linearly related to ICU stay time. The sepsis incidence of a patient who stayed for a long period in the ICU is higher compared to the previous stages.

In Figure 3.7, the architecture of the developed TASP model is displayed. The TASP model is a time-dependent combination of three sub-models. These sub-models consist of two tree-based methods and an RNN-based method, that are trained on different data parts with different feature sets.

Each hourly measured data from the training data contains an officially defined label. When a new hourly measurement of an unseen ICU stay is given, the TASP model will select the corresponding sub-model to make a sepsis prediction. For each of the earlier mentioned time-related stages, a sub-model is trained. The different sub-models are briefly discussed below.

M1: Early-stage sub-model

An independent tree-based model is trained on the first 9 hours of all patients to predict the sepsis risk in the early stage. For each case in the early stages, a set of features are created. There are 5 feature sets. Some of these feature sets are:

- F1: the original features for the specific hour
- F2: the min, max, mean, std, max-min of the 8 vital signs in the previous 6-hour window

The LightGBM is used to train the early-stage model using the 139 "created" features. The LightGBM framework is an efficient gradient boosting tree which has been widely used for classification and regression problems. Because LightGBM is less susceptible to overfitting, the forward filling method is used to impute missing values [23].

M2: Middle stage sub-model

Since the data size is much larger than the data of the early stage, two new sliding windows of 12 and 24 hours are added to the existing dataset. Besides, the addition of the sliding windows, the number of feature measurements are added to the features set. This represents the measurement frequency. Similar to the early stage, the missing values will be imputed using the forward filling method. Since this stage has a low sepsis rate, the weights for positive instances are increased. In contrast to the early stage, this model is trained on the complete data, rather than on the data of the middle stage. This allows the model to benefit from the information of the other stages [23].

M3: Late-stage sub-model

In this study, a Recurrent Neural Network (RNN) is used for the sepsis prediction in this late stage. A Recurrent Neural Network is a good framework to capture the transitory relations in sequential data. Since this has learnable decay rates, a Gated Recurrent Unit with a Decay (GRU-D) can measure the influence of the missing parts for final prediction. For each hour, the GRU-D can generate a hidden layer using the original 39 features. After the hidden layer is created, it is combined with the earlier constructed feature set. This is followed by a multi-layer perceptron which is used to make an hourly prediction for the onset of sepsis.

The two tree-based sub-models used for the early and middle stages are implemented using the LightGBM framework. The used parameters for this framework are a learning rate of 0.01, number of leaves = 70 and a minimum of 1000 instances in each leaf. The RNN sub-model for late-stage has the following core parameters. It has a hidden layer size of 39 and uses a learning rate of 0.0005. Most of the parameters in the three sub-models utilize the default parameters values. For each sub-model, 10-fold cross-validation is used to determine the best cut-off for the provided training dataset.

3.2.5 Utilizing Informative Missingness for Early Prediction of Sepsis

In contrast to the previously discussed research, this paper [24] focuses on the informative missingness of the dataset for predicting sepsis.

For this purpose, this research developed an XGBoost-based model for the early prediction of sepsis. Because the dataset shows certain characteristics, a naive model, which is only based on previous observations, is therefore not suitable for this task. The research proves the existence and importance of patterns in the missing data (informative missingness) and uses this to develop a more accurate model.

An initial study of the dataset was conducted, from which it was concluded that there is variation in ICU length of stay (ICU LOS) between non-sepsis and sepsis patients. The majority of patients who were not diagnosed with sepsis spent less than 60 hours in the ICU (with a mean LOS of 37 hours and a standard deviation (std) of 15.8 hours). Patients with Sepsis, on the other hand, spend more time in the ICU (a mean LOS of 60 hours with a standard deviation of 59.2 hours). This may be related to increased complications in treatment due to sepsis.

For this research, a model based on the XGBoost algorithm was implemented for sepsis prediction. The data will undergo pre-processing steps, where additional features will be created and added to the original dataset, which are displayed in Figure 3.8.

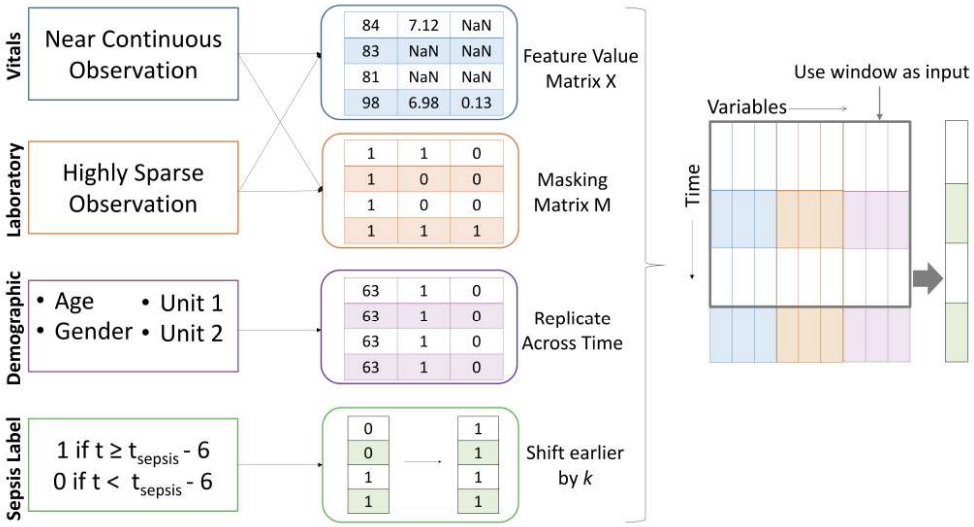

Figure 3.8: Data pre-processing steps for model with IM [24]

The dataset is a series of records, where each observation can be denoted as $x_t \in \mathbb{R}^d$, where t is the time of observation, x is the observation and d is the number of features. Because XGBoost is not a sequence learned model, a non-overlapping window of (w) hourly measurements is created and will be used as a single input. To forecast the Sepsis Label at time t, the records from x_{t-w+1} to x_t consisting of $d * w$ features are concatenated. The sliding window is moved by 1 hour to incorporate new hourly data. The use of sliding windows results in that each hourly measurement is used multiple times as input.

Each observation X_t is used as an input without missing data imputation. This means that the missing values are represented by NaN. A masking vector is generated to represent missing value patterns, where it is equal to zero indicates that a feature value is missing at time t and equal to one indicates that the feature value is present at time t.

The Utility Score metric was designed to predict sepsis up to six hours before $t_{optimal}$ and twelve hours before t_{sepsis}, resulting in a positive utility score. Since a non-sequential model and a limited window size was used, it is possible to shift further back to promote an earlier forecast. For Sepsis patients, this means that SepsisLabel = 1 if t > $t_{sepsis-(6+k)}$ and SepsisLabel = 0 if t < $t_{sepsis-(6+k)}$. An experiment was conducted for different values of K, and it was found that k = 3 achieves the highest utility score on the training data [7].

Before the XGBoost model is developed, 20% of the training data is randomly (patient-wise) selected and will be used to evaluate the performance of the final result. The best hyperparameters are found using 5-fold cross-validation on the other 80% of the training data.

3.3 Importance of early prediction for the medical sector

Sepsis can damage healthy tissues by spreading locally. In the worst case, it can even form a blood clot and restrict the blood flow to vital organs, resulting in organ failure and eventually death. With timely identification and treatment, sepsis is curable. Antibiotics and intravenous fluids administration should start immediately after diagnosis. Research has shown that for every hour of delay in the administration of antibiotics, there is an associated 6% rise in mortality [2], [3].

Even with highly developed medicine and advanced technology today, the mortality rate remains between 25% to 30% for severe sepsis and 40% to 70% for septic shock. Annually, twenty per cent of the in-hospital deaths are caused by sepsis [1]. Earlier detection can help mitigate the risk of sepsis. The earlier it is detected, the earlier the treatment can begin, the less harm is done to the body and the higher the chance of survival. Early treatment is as crucial for the survival of a patient as early recognition. However, it is difficult to recognize sepsis even for doctors, let alone in the early stage. In an international survey investigating the views of clinicians, 86% of responders felt that the symptoms could easily be misattributed to other medical conditions [25]. Most of the complications arise due to delay caused by this conflict. In two recent studies [6], [12], researchers suggest that there will be increased mortality by septic patients who have experienced delays in their (antibiotic) treatment. This effect is even more serious for patients who are suffering from septic shock, where hourly delays were associated with a 4-8 % increase in mortality per hour [12].

When the typical symptoms of sepsis manifest, the disease has already progressed to a certain degree, and the patient's life is already in danger.

It will be much better if the sepsis can be predicted early, with which the cost of treatment, the impairment sepsis cause to the patient body, the pain the patient goes through, and the chance of death will all decrease. That is the trend to fight sepsis. As technology is developing, algorithms implemented with machine learning can be applied to predict sepsis based on vital signs. As technology developing, algorithms implemented with machine learning can be applied to predict sepsis based on clinical data. Currently, some algorithms to predict sepsis for patients in ICU have already come out [26], [27].

3.3.1 Currently existing state of the art

A machine learning-based algorithm, InSight [27], has succeeded in predicting sepsis four hours ahead using only six vital signs with gradient tree boosting. It outperforms the existing sepsis scoring system. The Area Under Receiver Operating Characteristics (AUROC) is a performance metric score for a machine learning model. It is calculated on the true positive and false negative predictions the model gives together with different threshold values. For example, if the probability is larger than or equal to the threshold value, it will be classified into the positive class, numerically one, or into the negative class, zero. AUROC ranges from zero to one and is used as a summary of the model skill. The higher it is, the better the prediction values match the actual values, and the more suitable the machine learning model is for this situation. InSight is the first sepsis screening system to exceed the AUROC score of 0.90 using only vital sign inputs [28].

Another method called Artificial Intelligence Sepsis Expert (AISE) can accurately predict the onset of sepsis in an ICU patient four to twelve hours before clinical recognition. This is a machine learning algorithm, which is capable of predicting severe sepsis onset up to 48 hours in advance using only readily available vital signs extracted from the existing patient electronic health records, which reached an AUROC score of 0.93 [29].

Not only in the early detection of sepsis but also in predicting mortality rate did machine learning-based algorithms show their competence and accuracy [30]. Nowadays, the AI Clinician, a computational model with reinforcement learning, which can dynamically give suggestions for the optimal treatment on sepsis patients in ICU, has come out and been validated [31].

Despite these promising achievements, it is still in the development stage and still has a long way to go. The selection of the most appropriate treatment needs the physician's experience and judgement, patient's will, physical condition and examination result, and profound knowledge for his/her medical history [32]. The inputs for an algorithm are always limited as opposed to a physician. Many clinicians remain reluctant to incorporate AI algorithms into their daily clinical practice due to the time, efforts and expenses it will cost [32]. Also, the different hospitals have different electronic medical systems. A model trained with the clinical data from the first hospital may not work well on the second hospital. There needs to be more improvement and customization before this technology can enter medical clinical sepsis management.

3.3.2 Sepsis scoring standards

3.3.2.1 *SIRS*

Systemic inflammatory response syndrome, also called SIRS, is a syndrome of the whole body goes into the inflammatory state. It can be caused both by infections and non-infectious diseases. SIRS always features fever or abnormally rapid breathing.

Table 3: Systemic inflammatory response syndrome [33]

Feature	Value
Temperature	< 36 °C or > 38 °C
Heart rate	> 90 beats per minute
Respiratory rate	> 20 breaths per min or PaCO2 < 32 mmHg (unit of pressure, equal to 4.3 kPa)
White blood cell (WBC)	< $4x10^9$/L, or > $12x10^9$/L, or the ratio of immature neutrophils (namely band cells or immature white blood cells) over all blood cells ≥ 10%

As shown above, SIRS has four criteria. When two or more are met, patients are diagnosed with SIRS. If there is acute organ dysfunction, it will be termed "severe SIRS". SIRS is closely related to sepsis, which was established for and once used for the definition of sepsis.

Three definitions for sepsis have been developed over the years. The first definition, Sepsis-1, was developed together with SIRS at the International Sepsis Definitions Conference in 1991 [34]. It was defined as infection or suspected infection leading to the onset of SIRS. Sepsis complicated by organ dysfunction was termed severe sepsis, which could progress to septic shock, defined as "sepsis-induced hypotension persisting despite adequate fluid resuscitation" [35].

The second definition, Sepsis-2, came out as a list of diagnostic criteria after a task force recognized the limitations of SIRS in 2001. To be diagnosed with sepsis under the Sepsis-2 definition, as with Sepsis-1, an individual must have at least 2 SIRS criteria and a confirmed or suspected infection [34]. The definitions of sepsis and septic shock remained unchanged for more than twenty years.

Sepsis-3 was proposed in 2016 [15]. It defines sepsis as life-threatening organ dysfunction caused by a dysregulated host response to infection. Only in this new definition, the use of SIRS was abandoned in the identification of sepsis.

Seen from the history of the defining sepsis, the relationship between SIRS and sepsis was once close. Although already abandoned, SIRS might be a good candidate for the features in the sepsis prediction algorithm.

3.3.2.2 *Sequential Organ Failure Assessment*

Sequential organ failure assessment (SOFA) is presented as a score to track the extent of the organ function and the decline of a patient's health during the stay in an intensive care unit (ICU) [36]. It is calculated as the sum of six different scores, each of which representing the respiratory, cardiovascular, hepatic, coagulation, renal, and neurological systems [37].

The tables in Figure 3.9 below show how the SOFA score is calculated. In the case where the physiological parameters do not match with any row, zero is given. In the case where more than one row is matched, the one with the most points will be picked. The final SOFA score is calculated by adding all the scores for each table up.

SOFA Scoring for Respiratory system		SOFA Scoring for Nervous system		SOFA Scoring for Cardiovascular system		SOFA Scoring for Liver		SOFA Scoring for Coagulation		SOFA Scoring for Kidneys	
PaO2/FiO2 [mmHg (kPa)]	SOFA score	Glasgow coma scale	SOFA score	Mean arterial pressure OR administration of vasopressors required	SOFA score	Bilirubin (mg/dl) [µmol/L]	SOFA score	Platelets×103/µl	SOFA score	Creatinine (mg/dl) [µmol/L] (or urine output)	SOFA score
≥ 400 (53.3)	0	15	0	MAP ≥ 70 mmHg	0	< 1.2 [< 20]	0	≥ 150	0	< 1.2 [< 110]	0
< 400 (53.3)	+1	13–14	+1	MAP < 70 mmHg	+1	1.2–1.9 [20-32]	+1	< 150	+1	1.2–1.9 [110-170]	+1
< 300 (40)	+2	–12	+2	dopamine ≤ 5 µg/kg/min or dobutamine (any dose)	+2	2.0–5.9 [33-101]	+2	< 100	+2	2.0–3.4 [171-299]	+2
< 200 (26.7) and mechanically ventilated	+3	6–9	+3	dopamine > 5 µg/kg/min OR epinephrine ≤ 0.1 µg/kg/min OR norepinephrine ≤ 0.1 µg/kg/min	+3	6.0–11.9 [102-204]	+3	< 50	+3	3.5–4.9 [300-440] (or < 500 ml/d)	+3
< 100 (13.3) and mechanically ventilated	+4	< 6	+4	dopamine > 15 µg/kg/min OR epinephrine > 0.1 µg/kg/min OR norepinephrine > 0.1 µg/kg/min	+4	> 12.0 [> 204]	+4	< 20	+4	> 5.0 [> 440] (or < 200 ml/d)	+4

Figure 3.9: Table for calculating the SOFA score

The SOFA scoring system helps to predict the clinical outcomes of critically ill patients. According to an observational study at an Intensive Care Unit (ICU) in Belgium, the mortality rate is at least 50% when the SOFA score is increased, regardless of the initial score in the first 96 hours of admission, 27% to 35% if the score remains unchanged, and less than 27% if the score is reduced [38].

3.3.2.3 *qSOFA*

The Quick SOFA, also qSOFA, is a simplified version of the SOFA scoring system introduced by the Sepsis-3 group in February 2016. It is applied as an initial way to identify a patient with high infection risk. The qSOFA simplifies the SOFA score drastically by considering only three simply measured clinical criteria. Thus, it can be repeated serially on patients much more easily and quickly than the SOFA score.

Table 4: qSOFA Scoring Table [37]

Assessment	qSOFA score
Low blood pressure (Systolic blood pressure ≤ 100 mmHg)	1
High respiratory rate (Heart rate ≥ 22 breaths/min)	1
Altered mentation (Glasgow coma scale < 15)	1

The final qSOFA score is calculated by adding the three points for each assessment up. Although it cannot convey as much information as SOFA, the presence of two or more points of qSOFA after the onset of infection is always associated with a greater risk of prolonged ICU stay or even death. Infected patients with this are also more prone to develop sepsis. It is recommended to use qSOFA as a simple bedside prompt to identify whether or not patients outside an ICU are likely to have sepsis from the Third International Consensus Definitions for Sepsis (Sepsis-3) [39].

3.3.3 The current state of sepsis scoring

Among all the score standards, SOFA scoring has been proved to be the best one. In the 2016 SCCM/ESICM evaluation for criteria of identifying sepsis, SOFA stood out as the winner and was recommended to assess the severity of organ dysfunction in a potentially septic patient. From another comparison of the predictive validity for mortality in sepsis patients, SOFA was not significantly different from the more complicated LODS (Logistic Organ Dysfunction System) but was generally better than SIRS and qSOFA [14].

Other studies have also indicated that SIRS is not an ideal marker for sepsis. In a recent study, they evaluated the presence of SIRS criteria in 109,663 patients with infection and organ failure [16]. From the result, 12% of patients were classified as having SIRS-negative sepsis. For the utility of predicting mortality in an ICU setting, an increase in SOFA score of two or more was related to a higher death rate. SOFA had greater prognostic accuracy.

Here, prognostic accuracy mainly refers to the survival rate and life expectancy. For SIRS or qSOFA, which one has a better prognostic accuracy of mortality is not yet clear and more research is required [40], [41].

4 MATERIALS

4.1 Dataset

The dataset used in this thesis is provided by PhysioNet Computing in Cardiology Challenge 2019 [7]. It contains over 60,000 ICU patients with up to 40 clinical measurement variables for each hour of a patient. Sepsis-3 clinical criteria are applied for the detection of sepsis onset.

Data is obtained from three geographically distinct U.S. hospital systems: Beth Israel Deaconess Medical Center, Emory University Hospital, and an unidentified hospital system [7]. Each of them is labelled as hospital system A, B, and C in the thesis. Data and sepsis labels for 40,336 patients from hospital systems A and B are publicly available to download[1]. Data and labels of 24,819 patients from hospital system A, B and C are inaccessible to the public. It is used as a test dataset provided by the challenge organisers to evaluate the performance of the algorithms submitted by the challenge participants. Since this thesis was initiated after the challenge had concluded, it was not possible to run our algorithm on the hidden test set which is only accessible to the challenge organisers. Therefore, in this thesis, only the publicly available data from the hospital system A and B is used for the dataset. There is a class imbalance in the data. Only 2932 patients (7.26%) of the 40,336 patients were diagnosed with sepsis. Below is the table summarizing the dataset.

Table 5: Summary of the shared dataset for hospital system A and B [7]

Hospital system	A	B
Number of patients	20,336	20,000
Number of septic patients	1,790	1,142
Sepsis prevalence	8.8%	5.7%
Number of rows	739,663	684,508
Number of entries	5,536,849	4,950,064
Density of entries	20.6%	19.1%

The dataset is composed of a combination of eight vital sign variables, 26 laboratory measurement variables, six demographic variables and the sepsis label denoted as “SepsisLabel”. SepsisLabel is the target variable that needs to be predicted by the designed algorithm. The utility score is calculated based on the given SepsisLabel, the SepsisLabel predicted by the algorithm we design and the time. The mechanism to calculate the utility score is introduced in depth in chapter 5.4.5. Table 6 contains detailed information for each of them [7].

[1] https://archive.physionet.org/users/shared/challenge-2019/

Table 6: Clinical variables [7]

	Row	Measurement	Description
Vital Signs (1-8)	1	HR	Heart rate (beats per min)
	2	O_2Sat	Pulse oximetry (%)
	3	Temp	Temperature (°C)
	4	SBP	Systolic BP (mm Hg)
	5	MAP	Mean arterial pressure (mm Hg)
	6	DBP	Diastolic BP (mm Hg)
	7	Resp	Respiration rate (breaths per min)
	8	$EtCO_2$	End tidal carbon dioxide (mm Hg)
Laboratory Measurement Values (9-34)	9	BaseExcess	Excess bicarbonate (mmol/L)
	10	HCO_2	Bicarbonate (mmol/L)
	11	FiO_2	Fraction of inspired oxygen (%)
	12	pH	pH
	13	PaO_2	The partial pressure of carbon dioxide from arterial blood (mm Hg)
	14	SaO_2	Oxygen saturation from arterial blood (%)
	15	AST	Aspartate transaminase (IU/L)
	16	BUN	Blood urea nitrogen (mg/dL)
	17	Alkalinephos	Alkaline phosphatase (IU/L)
	18	Calcium	Calcium (mg/dL)
	19	Chloride	Chloride (mmol/L)
	20	Creatinine	Creatinine (mg/dL)
	21	Bilirubin direct	Direct bilirubin (mg/dL)
	22	Glucose	Serum glucose (mg/dL)
	23	Lactate	Lactic acid (mg/dL)
	24	Magnesium	Magnesium (mmol/dL)
	25	Phosphate	Phosphate (mg/dL)
	26	Potassium	Potassium (mmol/L)
	27	Bilirubin total	Total bilirubin (mg/dL)
	28	TroponinI	Troponin I (ng/mL)
	29	Hct	Hematocrit (%)
	30	Hgb	Hemoglobin (g/dL)
	31	PTT	Partial thromboplastin time (s)
	32	WBC	Leukocyte count (count/L)
	33	Fibrinogen	Fibrinogen concentration (mg/dL)
	34	Platelets	Platelet count (count/mL)
Demographics (35-40)	35	Age	Age (years)
	36	Gender	Female (0) or male (1)
	37	Unit 1	Administrative identifier for ICU unit (medical ICU); false (0) or true (1)
	38	Unit 2	Administrative identifier for ICU unit (surgical ICU); false (0) or true (1)
	39	HospAdmTime	The time between hospital and ICU admission (hours since ICU admission)
	40	ICULOS	ICU length of stay (hours since ICU admission)
Outcome (1)	41	SepsisLabel	For septic patients, SepsisLabel is 1 if t ≥ $t_{sepsis} - 6$ and 0 if t < $t_{sepsis} - 6$. For nonseptic patients, SepsisLabel is 0

The public dataset contains a lot of missing values. Most of the vital signs, like heart rate and temperature, are easy to measure and monitor with medical equipment. For the patients in ICU, their vital parameters including blood pressure and heart rate are continuously measured and displayed. Therefore, these measurements are always available. However, some measurements like pH, HCO_3 and $PaCO_2$ need a blood test, especially for the WBC and Lactate. Other tests like urine test are also often ordered on patients. These laboratory tests cannot be performed hourly, and the results take a long time to come out. As a result, a lot of data for these laboratory measurements is missing. For many of them, more than 90% of the data is missing. Below is a figure for the entry density of the three hospital systems.

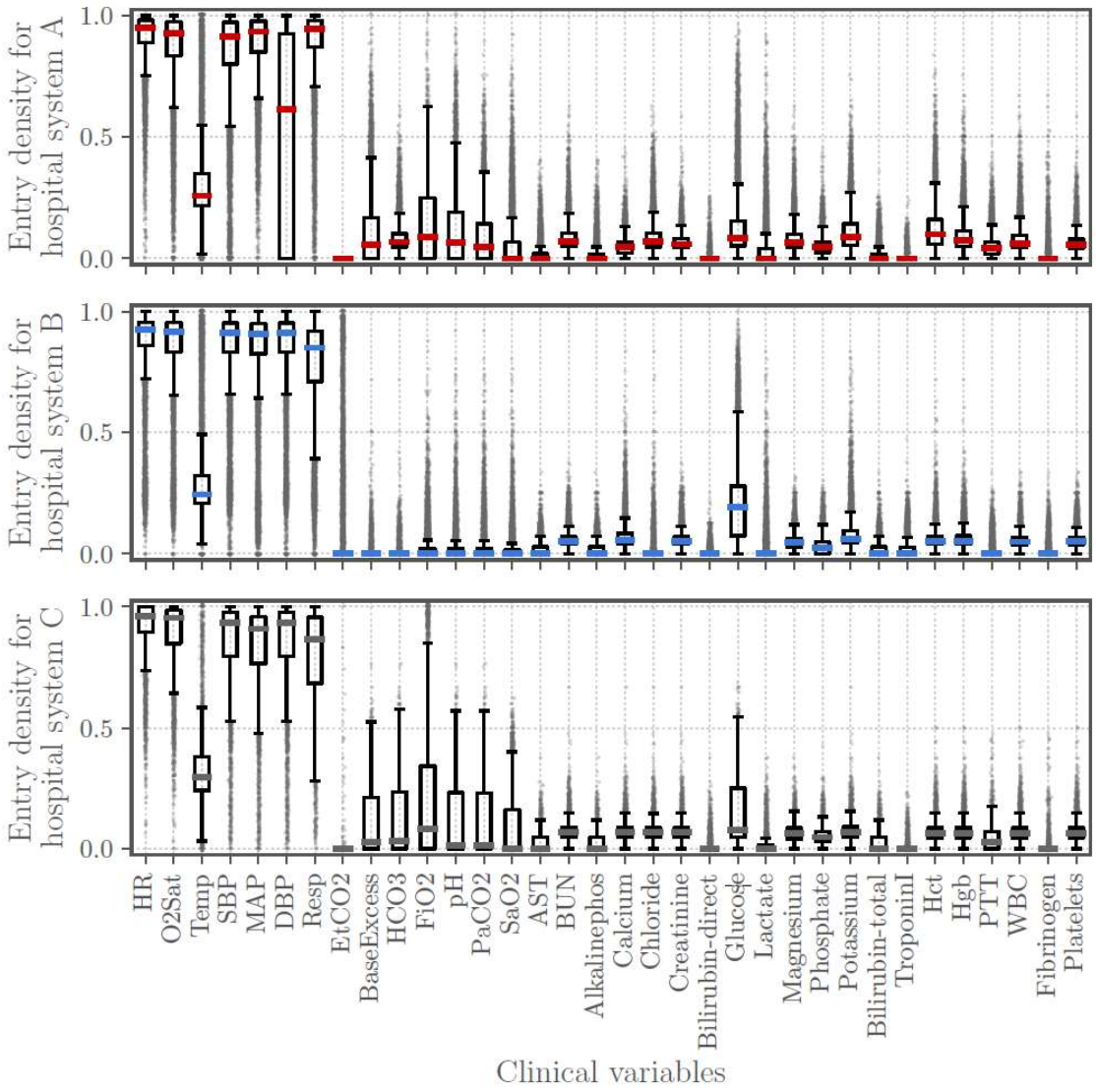

Figure 4.1: Entry density for A, B, and C hospital system [7]

4.2 Setup

The PhysioNet dataset was provided in a tabular form in text delimited files ('.psv' format) for each patient. For easy use of these '.psv' files, we opted to create several '.csv' files where these '.psv' files are combined for various sizes of patients. The data pre-processing and feature extraction were implemented based on previously executed researches. The advantage of using the Jupyter notebook framework is that the code can be written and executed in separate code blocks. The feature selection method and classification algorithms are implemented using the 'sklearn package' of python.

The implementation of the CUDA [42] and DASK [43] libraries are used to speed up the processing of the dataset. The DASK library was implemented for data analysis. From the conducted experiments, the DASK library provided the greatest speed improvement in the analysis. The DASK implementation ensures that the analysis is carried out over multiple processes instead of running on 1 core and 1 thread. Python was chosen for this research because it has a large library, and the developers do not need to concentrate on the technical complexities when solving a machine learning problem.

The dataset was processed using Python on different setups:

- Laptop with Windows 10 running on Intel i7-4510U and 16 GB of RAM
- Laptop with Windows 10 running on Intel i7-10875H and 16 GB of RAM
- Desktop with Ubuntu 20.04 running on Intel Xeon E3-1240 and 16GB of RAM
- KULeuven High-Performance Computing (HPC) server

Several experiments were conducted. Firstly, different CSV files ranging from 10 patients up to the full dataset of 40,366 patients were generated using the provided .psv files. For example, a .csv file (raw_data_100.csv) is generated, which contains the hourly measurement of 100 patients. Model fitting experiments were conducted on different datasets and classifiers. Personal laptops were used to execute these experiments for the smaller datasets (10-2000 patients). However, these were not powerful enough for experiments with the larger datasets (5000-40,366 patients). Therefore, the KULeuven HPC and desktop with Ubuntu are used to perform these operations. For the large dataset, these systems had a great benefit of the DASK implementation since the workload was distributed over multiple processes. The resources and services used in this thesis were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

4.3 Followed Course

At the start of this thesis, there was little experience with machine learning related principles. Hence it was decided to take an online machine learning course to get a better understanding of machine learning (ML) algorithms. This Massive Open Online Course (MOOC) course [44] was taught by Professor Andrew Ng, who is a professor of machine learning at Stanford University. This course provides a broad introduction to machine learning, data mining and statistical pattern recognition. It consists of supervised learning, unsupervised learning and best practices.

During this course, several applications of machine learning and advice for implementing ML algorithms are discussed. Examples are given of how learning algorithms can be used for analysing text (internet search, anti-spam), computer vision (image analysis), and also advice is given regarding how datasets can be split into different subsets. The advice for dataset splitting will be used in this thesis to validate the trained model. The course consisted of 11 chapters. For each of these chapters, exercises regarding the previously discussed theory needed to be conducted. Although the exercises were performed in MATLAB, the gathered knowledge provided a broad picture of the techniques behind a model.

5 METHODS

5.1 Missing data imputation

We follow an imputation strategy for handling missing data. It involves filling the missing entries by an educated guess before training the model. The missing value in the dataset is denoted as NaN, which stands for "not a number". In this thesis, five different filling methods are applied, each of which is introduced in this chapter. Below is the sorted table of the missing ratios of each column, showing the prevalence of each variable across the dataset.

Table 7: Missing ratio for each clinical variable for whole dataset

Column	Missing rate (%)	Column	Missing rate (%)
Patient_id	0.00	BUN	93.13
ICULOS	0.00	WBC	93.59
Gender	0.00	Magnesium	93.69
Age	0.00	Creatinine	93.90
SepsisLabel	0.00	Platelets	94.06
HospAdmTime	0.00	Calcium	94.12
HR	9.88	$PaCO_2$	94.44
MAP	12.45	BaseExcess	94.58
O_2Sat	13.06	Chloride	95.46
SBP	14.58	HCO_3	95.81
Resp	15.35	Phosphate	95.99
DBP	31.35	$EtCO_2$	96.29
Unit1	39.43	SAO_2	96.55
Unit2	39.43	PTT	97.06
Temp	66.16	Lactate	97.33
Glucose	82.89	AST	98.38
Potassium	90.69	Alkalinephos	98.39
Hct	91.15	Bilirubin_total	98.51
FIO_2	91.67	TroponinI	99.05
Hgb	92.62	Fibrinogen	99.34
pH	93.07	Bilirubin_direct	99.81

5.1.1 Zero filling

The zero filling strategy is the easiest way to impute the missing data. It is to fill all the missing data with zero. It is also the least computationally complex.

5.1.2 Mean filling

The mean filling strategy fills the missing data by the corresponding mean value of that entire column in the dataset. Firstly, without regard to the missing data, calculate the mean value for each column with the existing data from the original dataset. Then, fill in the missing data by its corresponding mean value. This method is also easy to implement. However, there is one shortcoming. If all the data for one column in the whole dataset is missing, there will be no mean value for this whole column. In this rare case, this whole column remains a column of missing values even after filling.

5.1.3 Forward filling

The forward filling method is also known as the Last Observation Carried Forward (LOCF). Each variable is filled with the previously observed value. LOCF is performed individually for each patient. It looks like filling the data in the forward direction.

Forward filling cannot fill the missing values in the data set completely. The missing data before the first valid value in one column of one patient remain missing. For a column with no valid value, it is also impossible to fill them in this way.

Only performing the forward filling is not enough. There are two ways to tackle this. The first one is to use zero filling to fill all the left missing values with zeros. This is referred to as the forward filling plus zero filling method. The second way is to calculate the mean of the observed training data from all the other patients and fill the missing values that cannot be filled in the LOCF method using this global training average. This is named the forward filling plus mean filling method.

5.1.4 Linear filling

Linear filling is performed for every column of every patient. Between two valid data in one column of one patient, it is assumed the measurement value is changing linearly with time. The missing values between these two data are linearly interpolated. That is why this method is called linear filling. For the missing values before the first valid value in one column of a patient, they are backwards filled with the first valid value. For the missing values after the last valid value, they are forwards filled with this last valid value.

For a column of one patient with no valid values, linear filling can do nothing to fill it. Similar to forward filling, the two same methods, zero filling or mean filling can be performed to solve this problem.

Unfortunately, the linear filling method fails to be a useable data imputation method. Filling the former missing value with future valid value is impossible in the real life. In real life, we can only estimate the missing measurement values for one patient by the current and former measurements. Besides, the deterioration of the patient can happen abruptly at any time in an hour. It is possible that the patient was non-septic one hour ago and is sepsis now. In this case, the measurement now is not a good estimation for the last hour.

5.1.5 KNN filling

The K-Nearest Neighbours (KNN) algorithm can not only be applied in classification, but also for data imputation. During data imputation, it replaces the missing value with the mean value of the K nearest neighbours found in the training set. Two samples are close if their features are numerically close except for the missing features. The distance of the two samples is defined as the Euclidean distance. For one sample, the distances from all the other samples are calculated. These samples are ranked by their distances. The first K samples with the smallest distance are considered to be the K nearest neighbours. To perform KNN, it is necessary to first split the dataset into training and test dataset. The KNN based Imputation model is trained by the training dataset and then performed on both training and test datasets.

KNN is the most computational complex one among all the data imputation methods. It needs to compute the distances of all the other samples to the current sample each time. As a result, it takes the longest time to run. Besides the long time it takes, the choice of K value is also a problem. The range of K can go up from one to the size of the dataset. What is the most optimal K value to give the best performance?

In the end, this costly method was not performed for the full dataset in the thesis.

5.2 Features augmentation

After imputation of the missing data in the training data, a prediction for the occurrence of sepsis can be made. However, there is no significant improvement in the sepsis prediction performance. To address these issues, research will be done to find methods that can improve the prediction performance. During the research of the best 5 PhysioNet Challenge papers, several methods were discovered which could be used to improve the prediction performance. These 'discovered' methods and one of our own will be briefly explained below. The used methods are:

- Columns with a high percentage of missing data are being removed
- Including medical criteria for sepsis
- Using a sliding window for the vital signs

5.2.1 Dropping columns

During the data analysis, it was discovered that the majority of the training data contains a lot of missing data. This data, which contains many missing data gaps, can not be used to predict the onset of sepsis. However, if the data with missing data were to be used anyway, this would lead to erroneous sepsis predictions. To address this problem, there can be opted to drop the columns of the dataset where a significant percentage of the data is missing. After this operation, the data significantly contains fewer missing data. This may lead to more accurate and reliable sepsis predictions. The downside of this method can be that the dropped columns contain variables useful for predicting sepsis, or that a great deal of related information may be removed.

5.2.2 SOFA, qSOFA, SIRS

The use of the existing medical sepsis evaluation criteria is a common method to augment the "sepsis" training features, is to use existing medical evaluation criteria. In chapter 3.3.2, the different existing evaluation criteria were discussed. The SIRS, qSOFA, and SOFA scoring methods are commonly used for medically determining a patient's occurrence of sepsis. The SIRS and qSOFA scores mainly use the vital signs of a patient. Since the vital signs have a low missing data score, this will lead to the fact that these "new" features will have a positive effect on the sepsis prediction of a patient. Consequently, the implementation of the SOFA score, which consists of a combination of vital signs and laboratory values, will not provide the expected improvement in the predicting performance. Therefore, mainly the SIRS and qSOFA score variables will be used to augment the existing patient features. Unfortunately, when the use of the medical indicators is combined with the dropping of the high missing data columns, it will lead to missing values which are necessary for the medical score indicators. As a result, it is not possible to use the qSOFA score sepsis indicator because one of the parameters used for the qSOFA score is missing in the "available" data.

5.2.3 Sliding Windows

Although the training and test datasets are split by patient ids, the data is put into the machine-learning model for each hour during the training process. For example, suppose there are 45 lines of data for one patient and each line contains the measurements and SepsisLabel for an hour. During the training, all data from all patients are shuffled and mixed up. The information contained in the data timely before the one line of data is lost. To extract longitudinal information from the time series, a sliding window approach is used.

Considering the missing ratio, vital signs except the $\mathrm{EtCO_2}$ are lightly missed. Comparing to the other laboratory measurement values, the missing ratios for vital signs range between 8% to 30%, which are much lower than 90%. Therefore, vital signs contain much more useful information and are suitable for the sliding window. Learning from the experience of the participants for the challenge, six is the best choice for the sliding window look-back size [21].

For each patient, the eight vital signs of the last hours are appended to each line of the dataset. Besides, the min, the max, the difference between the min and max, the mean, and the stand deviation of the eight vital signs for the last six hours, are all appended to the dataset. In this way, in total 48 new attributes are created and appended. For the record before six hours, a sliding window with as much size is used instead. For the first hour record of one patient, looking back sliding window is unable to perform. In this case, zero is used for all the sliding window attributes.

5.3 Machine learning

In the machine learning domain, learning algorithms can be divided into two main categories, namely supervised and unsupervised learning algorithms.

- Supervised learning: The algorithm is presented with sample inputs with the desired outputs. The goal of the algorithm is to learn a general rule that assigns inputs and outputs.
- Unsupervised learning: Compared to supervised learning, unlabelled data is provided to the learning algorithm. The algorithm must therefore look for similarities in the dataset and deduce a 'hidden' pattern that is present in the input data.

A supervised learning algorithm uses training data which consists of inputs combined with the right outputs. The algorithm will look for patterns in the data that corresponds with the desired outputs. A supervised learning algorithm can take in new unknown inputs after training and decide which mark the new inputs will be labelled as based on prior training results. A supervised learning model's goal is to predict the correct mark for newly presented input data. A supervised learning algorithm can be written in its most simple form as $y = f(x)$.

The variable y is the predicted output that is determined by a mapping function that assigns a class to an input value x. The function used to connect input features to a predicted output is created by the machine learning model during training. Supervised learning can be further split into two subcategories: Classification and Regression.

Regression algorithms predict a single numerical output value using training data. Examples of regression algorithms are Linear and Logistical regression.

Classification algorithms are used to predict the corresponding class/label belonging to the input data. These algorithms can be used for spam detection and character recognition and classification. Commonly used classification algorithms are Decision Tree, Random Forest and linear classifiers.

For this thesis, supervised machine learning methods will be used to predict the occurrence of sepsis. The reason for using supervised learning is that the provided training data is labelled with two classes: 0 or 1, which indicates whether or not a patient has sepsis. Therefore, unsupervised learning algorithms will not be used.

In machine learning, classification is a two-step process that consists of a learning step and a prediction step. The model is built in the learning phase using the provided training data. The model is used to predict the outcome of given data in the prediction stage.

After the training dataset was supplemented with the extra variables (SOFA, qSOFA, SIRS, Sliding window), a method is needed which will predict the possible occurrence of sepsis in a patient in the ICU based on the supplemented training data. To predict sepsis at an early stage, a model is needed that will be trained on the provided supplemented training data.

In the following chapters, the different machine learning models used in the experiments carried out will be discussed. These machine learning models were used based on the implementation of previous research regarding the early prediction of sepsis in ICU patients.

5.3.1 Decision Tree

The decision tree model is a machine learning model that can be used for both classification and regression problems. In this research, this model will be used to solve the classification problem of "correctly" labelling sepsis patients. The decision tree model creates a model in the form of a tree structure. The training data will be broken down into smaller subsets using a set of if-then-else decision rules. The result of the model is a tree with leaf nodes and decision nodes. A decision node has at least two possible outputs. The leaf nodes represent the possible classifications of the dataset [45]. For this research, the leaf nodes will either be 0 or 1, indicating the sepsis positiveness of a patient.

The decision tree model is a simple model that requires few computational resources and can handle irrelevant features in the dataset.

Like most other classifiers, the Decision Tree Classifier [46] takes two input arrays, namely:

- An array X with shape (n_samples, n_features) that holds the training samples
- An array Y of labels with shape (n samples,) holding the classification labels for the training samples

After the model is trained on these two inputs, it can be used for predicting the classification of the sample. An example of using this model to predict output labels is shown in Figure 5.1.

```
X_train = [[0, 0], [1, 1]] # training features
Y_train = [0, 1] #  # training labels
X_test = [[2,1], [0, 1]]  # test features

model = tree.DecisionTreeClassifier(random_state=2)
model = model.fit(X_train, Y_train) # training mode using the training dataset
model.predict(X_test) # predicting the output labels of the test dataset
```

Figure 5.1: Example of Decision Tree Classifier model usage in python using sklearn

5.3.2 XGBoost

XGBoost is the abbreviation for "Extreme Gradient Boosting". XGBoost is a decision tree-based ensemble machine learning algorithm that adopts the gradient boosting method [47]. Since XGBoost is based on the Gradient Boosting method, it augments an ensemble of "weak" classifiers to (create) a better performing 'final' model. However, XGBoost uses the base GBM framework optimized with some systems optimization and algorithmic enhancements. In Figure 5.2, the major XGBoost optimizations are shown.

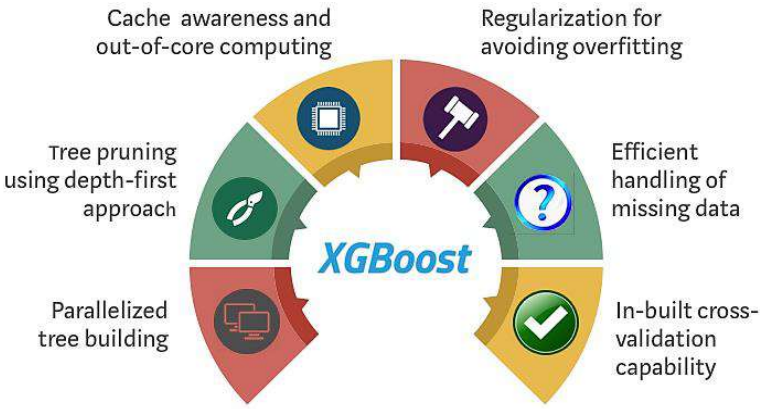

Figure 5.2: The optimization advantages of XGBoost [47]

XGBoost parallelize the process of creating sequential trees. This is possible thanks to the interchangeable loops used to build the base learners. Normally, an outer loop enumerates over the leaf nodes of the tree and a second inner loop calculates functions. These nested loops restrict parallelization. To improve runtime, the order of the loops is changed by initialization through global scanning of all instances and sorting using parallel threads. This switch improves the performance of the algorithm by compensating for all the overhead costs of calculation parallelization [47], [48].

XGBoost uses the 'max_depth' parameter instead of the criterion of the GBM framework and prunes the trees backwards. This depth-first method notably increases computational efficiency. To address the problem of overfitting when training a model, XGBoost uses regularization of the model which delivers better performance.

XGBoost admits sparse features for inputs by 'learning' the best missing value(s) depending on the training loss. This means that XGBoost can handle missing data.

XGBoost is an optimized parallelized version of the Gradient Boosting algorithm that uses tree-pruning, missing data handling and regularization to avoid overfitting.

The majority of the top five entries of the PhysioNet Challenge use the XGBoost classification model. This was an important reason to perform experiments with this model.

5.3.3 AdaBoost

AdaBoost is the abbreviation for Adaptive Boosting. It is an ensemble model that can be used in machine learning that can learn in the presence of class imbalance. It was the first successful binary classification boosting algorithm [49]. The AdaBoost classifier combines multiple low-performing classifiers into a strong classifier with high accuracy. The model is called adaptive boosting since the weights of the model are reassigned to each model with high weights for incorrectly classified data. Boosting is used in supervised learning to minimize variance and weak learners are transformed into a strong classifier. In Figure 5.3, the operation of the AdaBoost classifier is displayed. The AdaBoost method works as described in the following steps.

First, a training subset is selected at random. Next, the AdaBoost machine learning model is iteratively trained by choosing the training set based on the accuracy of the previous training. When data has been incorrectly categorized, then it will get a higher weight such that they have a higher chance of classification in the next iteration. These 'new' weights are assigned to the trained classifier in each iteration based on the classifier's accuracy. The classifier with the highest accuracy would be given the most weight. In other words, the models will be iteratively trained with information from the previous models. This method is repeated until the entire training data is fitted without error or when the limit of the number of estimators is reached. Finally, a 'vote' is performed across all of the built learning algorithms [49], [50]. The different outputs of the 'weak learners' are combined into a weighted sum that represents the final output of the boosted classifier.

The Python implementation of the AdaBoost is explained on the sklearn ensemble webpage [51].

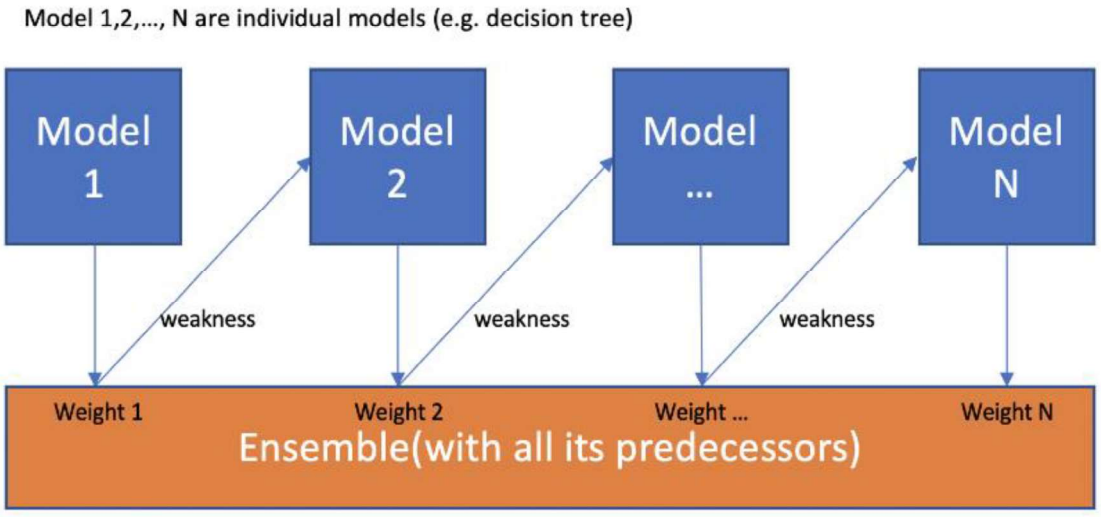

Figure 5.3: Graphical representation of AdaBoost method [50]

5.3.4 Gradient Boosting

Gradient boosting is a machine learning technique that produces a prediction model for regression and classification problems in the form of an ensemble of weak prediction models, usually decision trees. Boosting is the method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original dataset. When a decision tree is used as the weak learner, the resulting algorithm is referred to as gradient boosted trees. Like other boosting models, it builds the model in stages and generalizes it by allowing optimization of the loss function. It builds the model in stages, like other boosting models, and generalizes them by allowing optimization of every differentiable loss function. Gradient Boost has three main components [52], [53]:

- Loss function: The loss function's job is to estimate how well the model performs in making predictions with the given data. This can differ based on the nature of the issue.
- Weak Learner: A weak learner classifies data so badly that it is inferior to random guessing. Decision trees are the most common used for poor learners. However, other models may also be used in GBM.
- Additive Model: It is an iterative and sequential mechanism that involves applying decision trees one stage at a time. Each iteration could result in a decrease in the value of the loss function. A predetermined number of trees are added, or training is terminated when failure approaches an appropriate amount or no longer improves on an external validation dataset.

Gradient boosting is a greedy algorithm that can easily overfit a training dataset. As a result, regularization approaches are used to increase the algorithm's accuracy by reducing overfitting. Two methods that can be used to improve performance are discussed below.

Subsampling is used as the regularization process for GBMs. This improves the model's generalization properties while still reducing computing efforts. Randomness is introduced into the fitting protocol by subsampling. At each learning iteration, only a random subset of the training data is used to match a new base-learner. Without substitution, the training data is sampled.

Early stopping or tree pruning is an essential development that can be taken from the decision tree. This implies that if the ensemble model is limited by the number of trees. The overfitting would be avoided at the cost of the least amount of precision. Another observation is that the optimal number of boosts, at which the early stopping is considered, varies concerning the shrinkage parameter.

5.3.5 LightGBM

LightGBM or LGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient. Some of the advantages of LightGBM are [54]:

- Faster training speed and higher efficiency
- Lower memory usage
- Support of parallel, distributed and GPU learning
- Better accuracy

Most decision tree learning algorithms grow trees level-wise, row by row, shown in Figure 5.4. In contrast, LightGBM grows trees leaf-wise (best-first) as shown in Figure 5.5. Here the leaf with maximum delta loss will be chosen to grow. When holding the number of leaves fixed, leaf-wise algorithms will tend to achieve lower loss than level-wise algorithms.

When data is small, leaf-wise may cause over-fitting, so LightGBM includes the max_depth parameter to limit tree depth. However, trees still grow leaf-wise even when max_depth is specified. More information regarding LightGBM can be found on LightGBM documentation page [54].

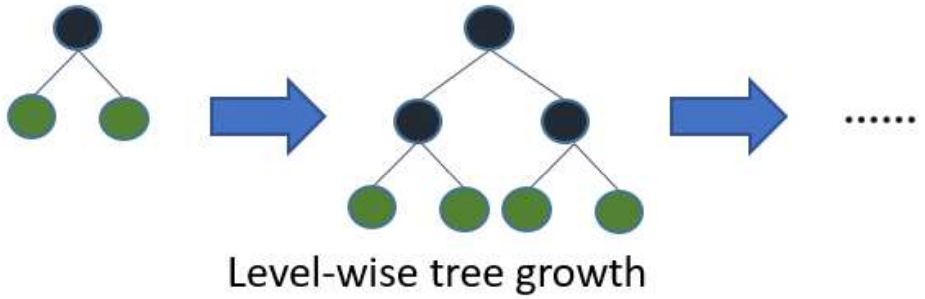

Figure 5.4: Level-wise tree growth [54]

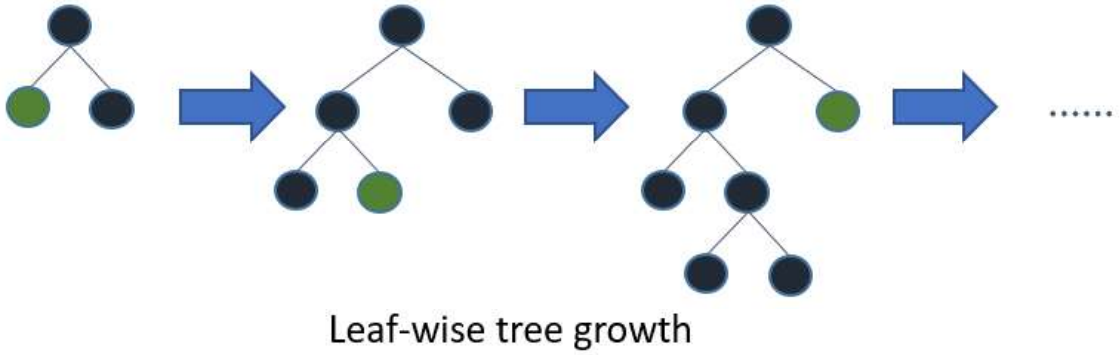

Figure 5.5: Leaf-wise Tree growth of LightGBM [54]

5.4 Evaluation

The algorithm is used to decide whether the patient in this hour is septic or not based on the measurement inputs. The output, SepsisLabel is labelled as either one, true or zero, false.

To evaluate the performance of different algorithms, five evaluation metrics are applied. Additionally, to make sure every data from the dataset is tested at least once, the K-Fold Cross-Validation is applied as a basic model for all the training with K =5. K-fold cross-validation splits the whole dataset into K consecutive folds. Then each fold is used once as a validation set while the remaining K-1 folds are used as the training set. In this way, the same algorithm is performed K times on the whole dataset, resulting in K scores for each evaluation metrics. The mean and standard deviation value of the K scores are calculated for each evaluation metric, which is used as the final performance measurement. Below is the introduction for each evaluation metric.

5.4.1 Accuracy

Accuracy is the most commonly used evaluation metric in machine learning. Since sepsis prediction is a binary classification problem, accuracy is very suitable here. It is obtained by comparing the prediction value with the actual value and calculating the ratio of the correct predictions among all the predictions. Ranging from 0% to 100%, a larger accuracy means more predictions match the actual value. Notwithstanding, accuracy is not a good evaluation method here. Over 95% of the actual value, the SepsisLabel, is zero. If the model is complete without any intelligence and just gives zero for all the predictions whatever the input is, it can still achieve an accuracy of more than 95%. That is the deficiency of accuracy. Only accounting for how many correct predictions is not sufficient.

5.4.2 F1Score

F1Score is more specific than accuracy. It is calculated by the harmonic mean of precision and recall. The precision is the number of true positive results divided by the number of all positive results. The recall is the number of true positive results divided by the number of all samples that should be predicted as positive. The chart below gives a better illustration of it.

As seen from Table 8, the prediction can be separated into four classes. The most important class that needs to predict precisely is the True Positive class. The most dangerous class is the False Negative class and must be avoided. The False Positive class also needs to be avoided, but it is not that severe as the False Negative class because at least the patient can survive in this case. The True Negative is not very important, since most of the SepsisLabel is negative. Precision and recall are two metrics to calculate the F1 score. The formula below shows how they work.

Table 8: Sepsis Binary classification

	True 1 (Sepsis)	**True 0 (Non-sepsis)**
Predict 1 **(Predict to be sepsis)**	True Positive (Good)	False Positive (Waste Resources)
Predict 0 **(Predict to be non-sepsis)**	False Negative (Deadly Dangerous)	True Negative (OK)

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad \text{Eq 5.1}$$

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad \text{Eq 5.2}$$

$$F_1 = \frac{2}{precision^{-1} + recall^{-1}} \quad \text{Eq 5.3}$$

Precision is better than accuracy in presenting the skills of the model. If no true positive prediction is made, the accuracy can still be as high as 95%, but the F1 score is 0. Only with a relatively high true positive prediction ratio can the F1 score be high.

5.4.3 AUROC

As has been briefly discussed in chapter 3.3.1, AUROC, the Area Under the Receiver Operating Characteristic, is calculated as the area under the Receiver Operating Characteristic (ROC) curve. A ROC curve is composed of the coordinate combinations of the True Positive Rate (TPR) and False Positive Rate (FPR) with different decision threshold values.

$$True\ Positive\ Rate(TPR) = recall = \frac{True\ Postive}{True\ Postive + False\ Negative} \quad \text{Eq 5.4}$$

$$False\ Positive\ Rate(FPR) = \frac{False\ Postive}{False\ Positive + True\ Negative} \quad \text{Eq 5.5}$$

The ROC curve starts at the left bottom corner (0, 0), where the decision threshold is one and both TPR and FPR are zero. ROC curve ends at the right top corner (1,1), where the decision threshold is zero and both TPR and FPR are one. The points between are obtained by calculating the TPR and FPR for different decision threshold. The curve is smoother with more points. The worst ROC curve is just a straight line from (0,0) to (1,1), which corresponds to a useless model with the AUROC score of 0.5. The ideal perfect ROC curve is composed of two straight lines. One goes from (0,0) to (0,1) and the other goes from (0,1) to (1,1). It means the TPR is always one, no False Negative predictions are made, and all the positive predictions made correctly match the actual case. This ideal situation achieves one for the AUROC. In real life, the AUROC is a value between 0.5 and 1. In most cases, an AUROC higher than 0.8 is considered to be excellent enough. Below is an example of the ROC curve.

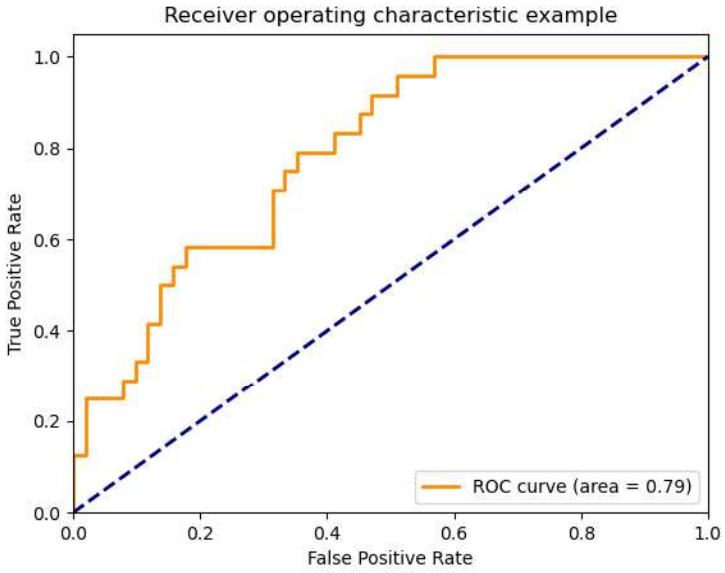

Figure 5.6: An example of AUROC

In conclusion, AUROC shows the competence of the model. It tells the model's ability to discriminate between positive and negative cases. The higher it is, the fewer False Negative predictions the model makes.

5.4.4 AUPRC

AUPRC is the area under the precision-recall curve. Like the ROC curve, a Precision-Recall (PR) curve is composed of the coordinate combinations of the precision and recall with different decision threshold values. The x-axis of a PR curve is the recall, and the y-axis is the precision.

The PR curve starts at the upper left corner (0,1), where the decision threshold is one. Every example is classified as negative. Thus, the recall is zero and precision is one. PR curve ends at the lower right corner (1,0), where the decision threshold is zero. All the examples are regarded as positive. Thus, the recall is one and precision is very low. The points between are obtained by calculating the precision and recall for different decision thresholds between zero and one. The curve is smoother with more points.

AUPRC shows the ability to find positive examples without marking negative examples as positive. It cannot surpass the fraction of the positive examples. Since less than 10% of the data is sepsis and AUROC is always larger than 0.5, AUPRC is much smaller than AUROC here. It is suggested to use AUROC when there are roughly equal numbers of observation for each class and to use AUPRC when there is a large class imbalance. Therefore, AUPRC is a more suitable evaluation metric than AUROC in this thesis [7].

5.4.5 Utility Score

The utility score is the optimal evaluation metric in sepsis prediction use case on the current hourly collected dataset. It takes not only correctness but also timeliness into consideration. Created by the PhysioNet Challenge, it rewards the early sepsis prediction and penalizes the late or missed sepsis predictions for sepsis patients. Sepsis predictions for non-sepsis patients are also penalized. The scores for different predictions are different. $U(s,t)$ is defined as a scoring function for a patient record at time t.

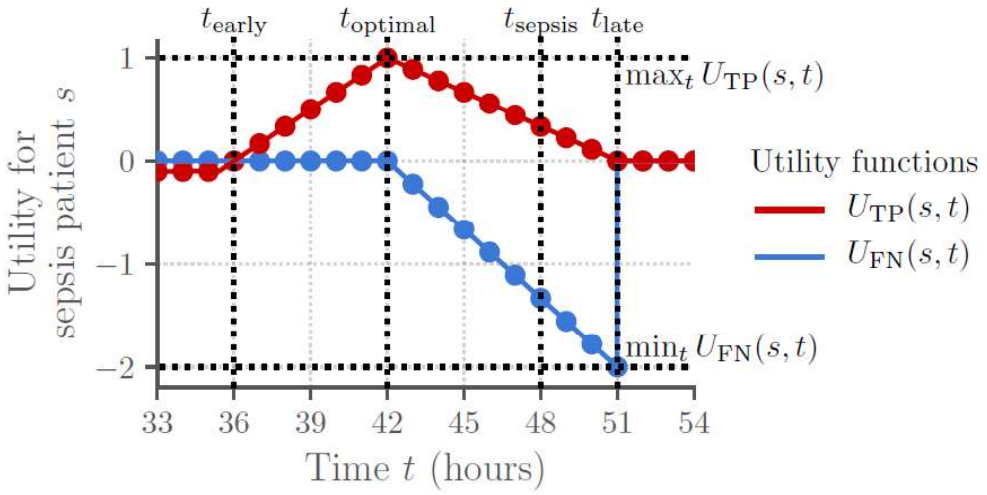

Figure 5.7: Clinical utility for septic patients [7]

$$U(s,t) = \begin{cases} U_{TP}(s,t), & postive\ prediction\ at\ time\ t\ for\ a\ septic\ patient\ s, \\ U_{FP}(s,t), & postive\ prediction\ at\ time\ t\ for\ a\ nonseptic\ patient\ s, \\ U_{FN}(s,t), & negative\ prediction\ at\ time\ t\ for\ a\ septic\ patient\ s, \\ U_{TN}(s,t), & negative\ prediction\ at\ time\ t\ for\ a\ nonseptic\ patient\ s, \end{cases}$$

$U_{TP}(s,t), U_{FP}(s,t), U_{FN}(s,t)$, and $U_{TN}(s,t)$ are four different variations of $U(s,t)$. They are illustrated in Figure 5.7 and Figure 5.8. Here it is supposed a patient has been in ICU for 33 hours.

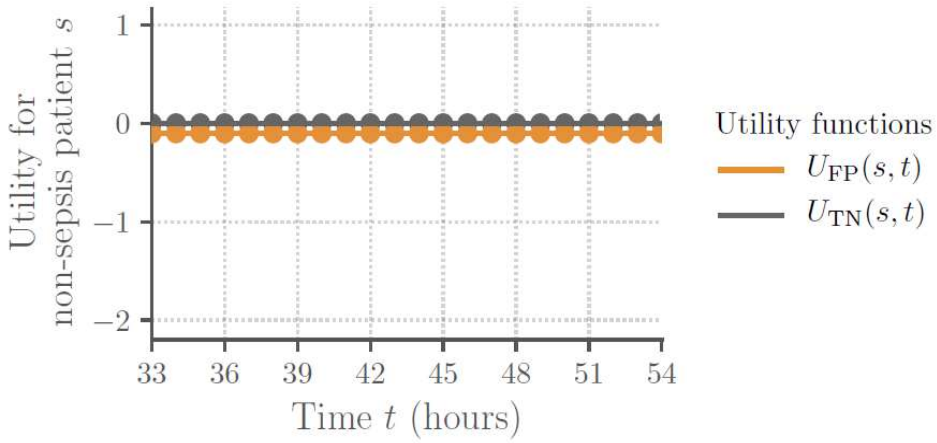

Figure 5.8: Clinical utility for non-septic patients [7]

In Figure 5.7, it is assumed that the patient develops sepsis at the 48th hour. Thus, t_{sepsis} equals 48. Early sepsis detection is beneficial, and the optimal time is six hours before the occurrence of sepsis, while late or missed septic predictions are misleading and harmful. However, too early prediction of sepsis may be implausible or unhelpful. It should not detect sepsis as early as more than 12 hours before the occurrence as it creates a false alarm that can cause alarm fatigue in the medical professional. Therefore, $t_{optimal}$ is 42 and t_{early} is 36. In the three hours after the sepsis occurrence, it's still acceptable for sepsis prediction since it's still not too late. That yields t_{late} to be 52. For $U_{TP}(s,t)$, sepsis prediction between t_{early} and t_{late} is rewarded. The most reward is at $t_{optimal}$ and the closer to the $t_{optimal}$, the more the reward is. Notwithstanding, sepsis prediction before t_{early} is slightly penalized. For $U_{FN}(s,t)$, non-septic prediction after $t_{optimal}$ is penalized. The later the non-septic prediction is, the more severe the penalization is. This means the algorithm is giving the wrong prediction, misleading the doctors, and delaying the treatment.

Figure 5.8 shows the score for the non-septic patients in their ICU stay. For $U_{FP}(s,t)$, false sepsis prediction contributes to alarm fatigue, lower confidence for the algorithm, antibiotic overuse and overall poor allocation of the hospital resources [7]. Although it is not as dangerous as the False Negative prediction, it is also unhelpful and should be avoided. That's why it is slightly penalized. For $U_{TN}(s,t)$, it remains to be zero. Non-septic prediction for non-septic is neither rewarded nor penalized.

Given all the predictions for all hourly time windows $T(s)$ for every patient record, the total score of an algorithm is defined as the sum of all the scores for all the predictions.

$$U_{total} = \sum_{s \in S} \sum_{t \in T(s)} U(s,t) \qquad \text{Eq 5.6}$$

For easier interpretability, the total score is normalized. The optimal algorithm with the highest score receives a normalized score of 1 and the completely inactive algorithm that only makes non-sepsis predictions receive 0. It is optimized by the equation below. The utility score is viewed as the determining evaluation metric in this thesis.

$$U_{normalized} = \frac{U_{total} - U_{no\ predictions}}{U_{optimal} - U_{no\ predictions}} \qquad \text{Eq 5.7}$$

5.4.6 Baseline

Baseline is not an evaluation metric for the prediction algorithm. We consider the ratio of negative examples to total examples as the baseline. It is more like setting a line for the accuracy to surpass. If the algorithm is lazy and only gives zero prediction, the accuracy is equal to the baseline. A good algorithm that can give correct positive and negative predictions should achieve an accuracy higher than the baseline.

6 EXPERIMENTS & RESULTS

The main experiments carried out during the master's thesis will be briefly discussed in the following chapters. The results and conclusions that can be drawn from these experiments will also be discussed. First, the experiments for the imputation of the missing data are discussed, followed by the conducted experiments for reliably predicting the occurrence of sepsis in a patient.

Along with all the experiments, different methods are compared for their influences on the algorithm's performance evaluated by the utility score. For different stages, the method that achieves the highest or increases the utility score is kept, while those that decrease the utility score are abandoned. In this way, the best performance with the highest utility score is reached at last.

6.1 Experiment for imputing the missing data

In chapter 5.1, the different methods that can be used to fill in the missing data in the provided training data were discussed. In the following chapters, the different experiments that were conducted to find the best filling method, in terms of performance (utility score, F1-score, accuracy), for imputing the missing data will be discussed.

6.1.1 KNN-filling method

The KNN filling method uses the values of the K nearest neighbours to fill in the missing data. For this method, we will experimentally investigate which value of K (number of nearest neighbours) achieves the highest performance scores for the different models. As previously mentioned in this thesis, the dataset training was performed using 10-Fold cross-validation. This means that the dataset is divided into 10 slices and the training data is evaluated for each of the 10 folds. Since the KNN method is computationally expensive, we tested the KNN based imputation approach by varying the number of patients from 400, 1000, 2000 and 5000 patients. For each dataset size, only the model that achieves the best performance will be selected. All the experiments were conducted using the following classifiers:

- Decision Tree
- XGBoost
- AdaBoost
- Logistic Regression
- Random Forest
- Gradient Boosting

6.1.1.1 *KNN on 400 patients*

For the 400 patient datasets, the experiments were conducted with the value of K (nearest neighbours) ranging from 5 to 370. For this range of K values, the range set was divided into two subsets with two different step sizes:

- In the first subset where K ranges between 5 and 250, a step size of 5 was used
- The second subset, where K ranges from 250 up to 370, is iterated using a step size of 15

The models mentioned above in the intro of the KNN-filling chapter will be used to perform this experiment. The Decision Tree classifier achieved the overall highest mean utility score for the KNN filling experiment.

After results comparison of the different classifiers, the Decision Tree classifier provides the overall best performance. It achieves the highest utility score of all the used classifiers and the Logistic Regression classifier performs the worst of them all. In Table 9, a slice of the results of the experiments and the best values of K for the KNN filling method for 400 patients is shown. In Figure 6.1, an error bar graph of the Decision Tree classifier results of the KNN filling method is shown. From this graph, it can be concluded that there is a lot of deviation in the utility score for a small, 400 patients, dataset. In Table 9, the best scores are displayed in bold. The highest mean utility score is achieved when K is equal to 325, the F1Score_mean is the highest at K equals 250 and finally, the highest mean accuracy score is achieved at K= 90.

Table 9: Results of KNN filling experiment with Decision Tree classifier for the 400 patients dataset

Decision Tree Classifier						**Best metrics**		
K	**5**	**10**	**15**	**20**	**25**	**90**	**250**	**325**
UtilityScore_mean	0.138	0.184	0.148	0.239	0.123	0.166	0.195	**0.275**
UtilityScore_std	0.264	0.267	0.274	0.248	0.297	0.271	0.226	0.275
F1Score_mean	0.123	0.144	0.127	0.140	0.111	0.126	**0.184**	0.158
F1Score_std	0.099	0.098	0.114	0.085	0.126	0.109	0.126	0.093
AUROC_mean	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
AUPRC_mean	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Accuracy_mean	0.936	0.934	0.941	0.939	0.930	**0.948**	0.933	0.924
Accuracy_std	0.047	0.058	0.041	0.032	0.060	0.023	0.057	0.052
Baseline_mean	97.875	97.875	97.875	97.875	97.875	97.875	97.875	97.875
Total time (sec)	1158.48	1157.86	1151.64	1149.13	1308.18	1219.30	1184.300	1209.38

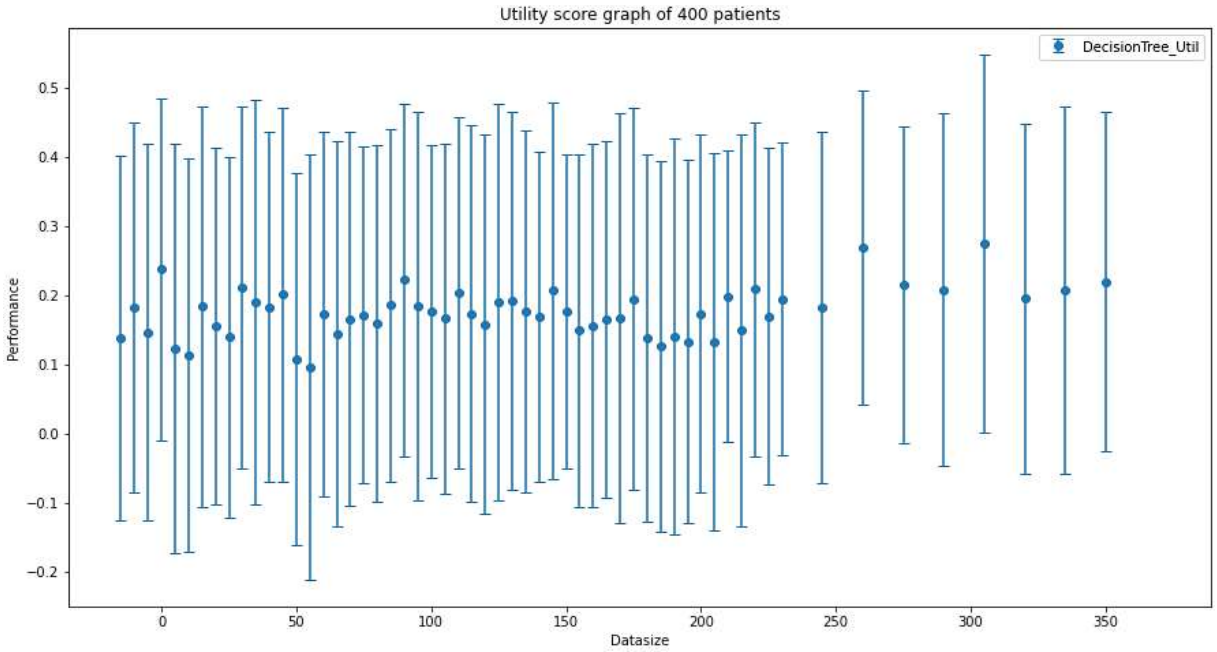

Figure 6.1: Error bar graph of the utility score of Decision Tree classifier for the 400 patients dataset (KNN)

6.1.1.2 *KNN on 1000 patient*

Similar to the 400 patient dataset experiment, the experiments were also conducted with the value of K (nearest neighbours) ranging from 5 to 370. For this range of K values, the range set was divided into two subsets with two different step sizes.

- In the first subset where K ranges between 5 and 250, a step size of 5 was used
- The second subset with K values ranging from 250 up to 370, is iterated using a step size of 15

The models mentioned at the beginning of the KNN-filling chapter will be used to perform these experiments.

Similar to the KNN missing data filling experiment for 400 patients, here the Decision Tree classifier also achieves the highest utility score. However, the Random Forest classifier achieves the same highest utility score as the decision tree. But the average utility score of the Random Forest classifier is lower than that of the decision tree classifier. A slice of the results of the KNN filling experiment using the decision tree classifier is shown in Table 10. In Figure 6.2, an error bar graph of the utility scores with corresponding standard deviations of all the different values of K is shown. Here, it can be noted the standard deviation is still larger by the executed experiments using the 1000 patients dataset. The best scores are displayed in bold in Table 10. Here, there can be observed that the highest mean utility score is achieved when K is equal to 210, the highest F1Score_mean is achieved at K equals 355 and finally when K= 325 the highest mean accuracy score is achieved.

Table 10: Results of KNN filling experiment with Decision Tree classifier for the 1000 patients dataset

Decision Tree Classifier						Best metrics		
K	**5**	**10**	**15**	**20**	**25**	**210**	**325**	**355**
UtilityScore_mean	0.126	0.097	0.090	0.113	0.134	**0.152**	0.133	0.137
UtilityScore_std	0.061	0.098	0.091	0.088	0.118	0.092	0.062	0.098
F1Score_mean	0.125	0.112	0.091	0.100	0.106	0.126	0.123	**0.127**
F1Score_std	0.047	0.076	0.055	0.054	0.050	0.044	0.052	0.069
AUROC_mean	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
AUPRC_mean	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Accuracy_mean	0.953	0.951	0.947	0.944	0.943	0.953	**0.959**	0.952
Accuracy_std	0.020	0.014	0.025	0.020	0.028	0.019	0.017	0.019
Baseline_mean	98.272	98.272	98.272	98.272	98.272	98.272	98.272	98.272
total time (sec)	1801.29	1792.05	1798.99	1801.52	1798.23	1842.36	1886.65	1874.15

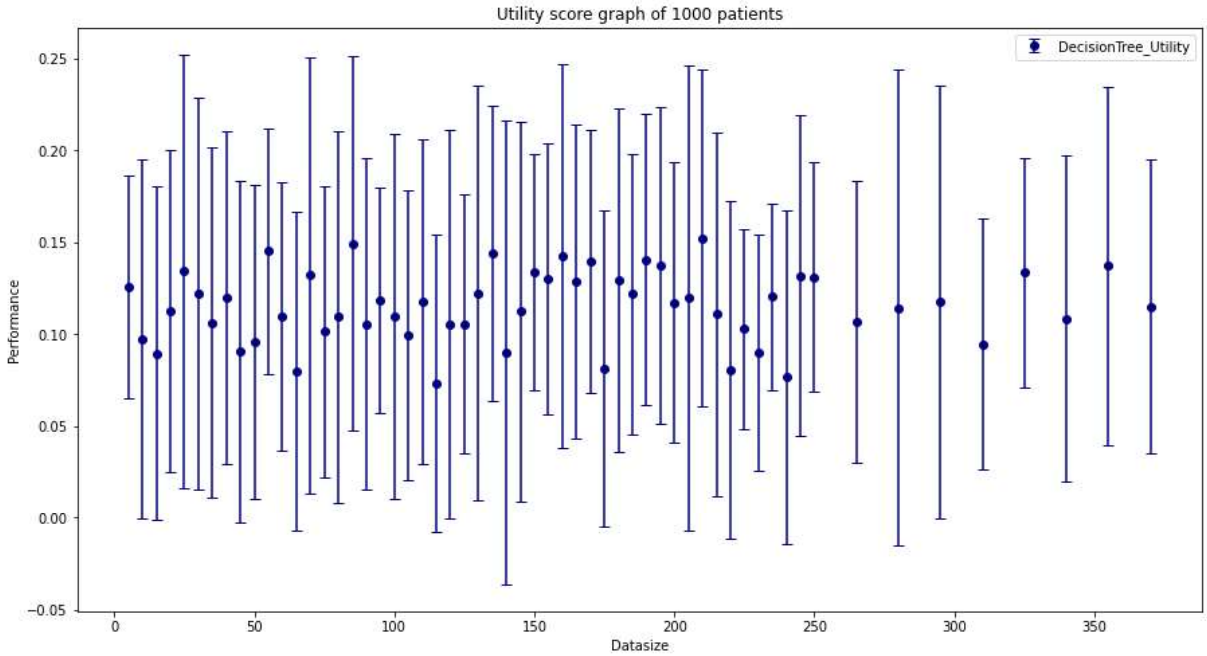

Figure 6.2: Error bar graph of the utility score of Decision Tree classifier for the 1000 patients dataset (KNN)

6.1.1.3 *KNN-filling on 2000 & 5000 patients*

From the experiments for the KNN wise filling of the missing data in the 400 and 1000 patient dataset, it can be deduced that the time needed to calculate these missing data increases significantly. Therefore, for the 2000 and 5000 patient datasets, it was decided to only perform experiments of the K values that gave the best results in the previous experiments. In this case, the K value 210 and 355 will be used, which gave the best results in terms of util score in the previous experiments. The results of these KNN-filling experiments are displayed in Table 11 and Table 12. From these tables, it can be observed that the Decision Tree classifier achieves a utility score of 0.1349 when using the 210 nearest neighbours. However, when this experiment is performed for the 5000 patients dataset with the same number of close neighbours and the same classifier, it only achieves a utility score of 0.0711.

Table 11: Results of KNN-filling on the 2000 patients dataset

Decision Tree Classifier		
K	**210**	**325**
UtilityScore_mean	**0.135**	0.127
UtilityScore_std	0.072	0.078
F1Score_mean	0.121	0.106
F1Score_std	0.045	0.039
AUROC_mean	0.000	0.000
AUPRC_mean	0.000	0.000
Accuracy_mean	0.951	0.944
Accuracy_std	0.010	0.016
Baseline_mean	98.162	98.162
Total time (sec)	18964.78	20363.62

Table 12: Results of KNN-filling on the 5000 patients dataset

Decision Tree Classifier		
K	**210**	**325**
UtilityScore_mean	**0.071**	0.071
UtilityScore_std	0.033	0.034
F1Score_mean	0.083	0.082
F1Score_std	0.020	0.020
AUROC_mean	0.000	0.000
AUPRC_mean	0.000	0.000
Accuracy_mean	0.953	0.951
Accuracy_std	0.009	0.005
Baseline_mean	98.171	98.171
Total time (sec)	47795.27	48472.38

Since the time required to complete an experiment on the dataset of 5000 patients is very long, it is not practically feasible to apply this method to the entire dataset. This thesis aims to predict sepsis for a patient in a timely manner. However, this is not possible to use the KNN filling method because it takes a long time to impute the missing data. As mentioned in chapter 3.3, each hour delay in treatment results in a 4-8% increase in mortality. For this reason, the KNN-filling method experiment was stopped prematurely. The aim was to make a sepsis prediction for a patient as "quickly" and reliably as possible. This means that KNN will not be used as the missing data imputation method in the final algorithm for reliable predicting the onset of sepsis.

6.1.2 Linear filling

As previously stated in chapter 5.1.4, the missing data will be filled linearly with the available data. Since the provided data consists of medical data from ICU patients regarding the time they spent in the ICU. The linear forward filling method will use both the past and "future" values to impute the missing data. In a hospital, this filling method can not be used since they only have a patient's past medical record and no future medical data. The goal is that the developed algorithm can be used in hospitals. As a result, it was determined not to perform any experiments for this filling process.

6.1.3 Zero filling

The functionality of this filling method is discussed in chapter 5.1.1. During this filling method experiment, the missing values will be replaced with the value zero. This and the following experiments were performed on different datasets with the following classifiers:

- Decision Tree (DT)
- XGBoost (XGB)
- AdaBoost (ADA)
- Logistic Regression (LR)
- Light-GBM (LGBM)
- Gradient Boosting (GRAD)

For the smaller, 400 and 1000 patients, dataset, the results of all the classifiers will be displayed. However, for the larger datasets (5000 patients, training dataset A) only the results of the two best performing classifiers will be shown. In Table 13 and Table 14, the results of the zero filling experiments on the 400 and 1000 patients dataset are shown. From these tables, some observations can be made. For example, it can be deduced that AdaBoost achieves the highest utility score for the 400 patients dataset. But for the 1000 patients dataset, it achieves a worse utility score and has not even one of the best utility score of the 1000 patient dataset experiment. Secondly, it can also be derived that the Decision Tree classifier achieves a relatively high utility score for both dataset experiments. This will surely be one of the classifiers that will be used for experiments on larger datasets.

Table 13: Results of zero filling experiment on the 400 patients dataset

400 patients	DT	XGB	ADA	GRAD	LGBM
UtilityScore_mean	0.174	0.120	**0.210**	0.067	0.021
UtilityScore_std	0.282	0.172	0.280	0.097	0.065
F1Score_mean	0.152	0.108	0.175	0.097	0.016
F1Score_std	0.132	0.159	0.146	0.149	0.049
AUROC_mean	0.000	0.751	0.793	0.759	0.724
AUPRC_mean	0.000	0.14	0.240	0.159	0.074
Accuracy_mean	0.929	0.943	0.952	0.948	0.975
Accuracy_std	0.059	0.060	0.059	0.058	0.012
Baseline_mean	97.880	97.880	97.880	97.880	97.880
Total time (sec)	11.840	40.750	40.670	48.390	91.320

Table 14: Results of zero filling experiment on the 1000 patients dataset

1000 patients	DT	XGB	ADA	GRAD	LGBM
UtilityScore_mean	**0.154**	0.136	0.114	0.114	0.007
UtilityScore_std	0.115	0.078	0.082	0.090	0.022
F1Score_mean	0.109	0.126	0.129	0.123	0.013
F1Score_std	0.063	0.067	0.079	0.091	0.030
AUROC_mean	0.000	0.777	0.807	0.783	0.758
AUPRC_mean	0.000	0.123	0.134	0.110	0.087
Accuracy_mean	0.938	0.966	0.971	0.968	0.980
Accuracy_std	0.028	0.017	0.012	0.020	0.008
Baseline_mean	98.260	98.260	98.260	98.260	98.260
Total time (sec)	19.140	64.380	87.510	43.030	164.640

Following the results of the 400 and 1000 patients dataset experiment, the best performing classifier of these experiments will be used for experiments on larger datasets. In Table 15, the results of the best performing classifier for experiments on the 5000 patients data and the "full" training dataset A. From both experiments, it can be observed that the Decision Tree and XGBoost classifier perform well. The Decision Tree classifier achieves a utility score of 0.1015 and 0.094 for the 5000 patients and training dataset A. The performance of this classifier (DT) does not decrease that much when the dataset increases. In contrast, the XGBoost classifier achieved utility score decreases considerably. The utility on the training A dataset is about half of the utility score on the 5000 patients dataset. As a result, when opting for the zero filling method as the imputation method for missing values, combined with this, it is recommended to use the Decision Tree classifier for a larger dataset, because it will more than likely give the 'best' utility score on larger datasets.

Table 15: Results of the zero filling experiment on larger datasets

Dataset size	5000		Set_A	
Zero filling	**DT**	**XGB**	**DT**	**XGB**
UtilityScore_mean	**0.102**	0.073	**0.094**	0.046
UtilityScore_std	0.034	0.038	0.013	0.010
F1Score_mean	0.099	0.100	0.100	0.080
F1Score_std	0.017	0.033	0.010	0.014
AUROC_mean	0.000	0.734	0.000	0.764
AUPRC_mean	0.000	0.073	0.000	0.089
Accuracy_mean	0.947	0.972	0.946	0.973
Accuracy_std	0.011	0.005	0.004	0.003
Baseline_mean	98.180	98.180	97.800	97.800
Total time (sec)	100.070	450.040	398.900	1705.000

6.1.4 Mean filling

In this set of experiments, the missing data was filled with the mean value of the training dataset. As previously mentioned in this thesis, the dataset is trained using 10-Fold cross-validation and therefore the dataset will be filled with the mean value of the training data. These mean filling experiments were conducted for previously mentioned classifiers using different dataset sizes ranging from 400 patients up to the full data. For mean filling and the other filling methods, only the result of the classifier which achieves the highest utility score will be discussed. The detailed results of the conducted experiments can be found in the annexe. From the results of the 400 and 1000 patients dataset experiment shown in Table 16 and Table 17, there can be observed that the Decision Tree classifier performs overall the best compared to the other classifiers mentioned in the tables. Unfortunately, the Logistic Regression classifier is not mentioned in this tables since it underperformed compared to the other classifiers.

Table 16: Results of mean filling experiments on 400 patients dataset

400 patients	DT	XGB	ADA	GRAD	LGBM
UtilityScore_mean	**0.171**	0.100	0.064	0.123	0.051
UtilityScore_std	0.266	0.163	0.108	0.109	0.082
F1Score_mean	0.136	0.114	0.061	0.128	0.077
F1Score_std	0.098	0.176	0.089	0.112	0.139
AUROC_mean	0.000	0.742	0.735	0.788	0.749
AUPRC_mean	0.000	0.130	0.147	0.213	0.147
Accuracy_mean	0.936	0.945	0.969	0.948	0.952
Accuracy_std	0.056	0.059	0.021	0.056	0.056
Baseline_mean	97.880	97.880	97.880	97.880	97.880
Total time (sec)	16.940	43.900	13.410	28.400	38.970

Table 17: Results of mean filling experiments on 1000 patients dataset

1000 patients	DT	XGB	ADA	GRAD	LGBM
UtilityScore_mean	**0.157**	0.124	0.053	0.098	0.117
UtilityScore_std	0.133	0.093	0.059	0.076	0.065
F1Score_mean	0.119	0.112	0.077	0.104	0.116
F1Score_std	0.081	0.079	0.089	0.068	0.063
AUROC_mean	0.000	0.757	0.777	0.793	0.772
AUPRC_mean	0.000	0.103	0.126	0.106	0.123
Accuracy_mean	0.938	0.965	0.980	0.971	0.963
Accuracy_std	0.022	0.014	0.005	0.013	0.019
Baseline_mean	98.26	98.26	98.26	98.26	98.26
Total time (sec)	19.4	61.05	29.22	97.66	75.53

In Table 18, the results of the mean filling experiment are displayed. The experiments were performed on the 2000 and 5000 patients dataset and the provided training set A. For each of the dataset, only the two best performing classifiers in terms of utility score are shown. What is striking in this table is that the Decision Tree classifier always achieves the best performance. But as the dataset gets larger, the performance of the XGBoost classifier increases. For the experiment on the 5000 patients dataset, the XGBoost classifier performs marginally worse than the Decision Tree classifier. But the experiment conducted on the full training set A, the performance of XGBoost decreases dramatically. It obtained a utility score that is nearly half of the utility score achieved by the Decision Tree classifier.

Table 18: Result of mean filling on the larger datasets

	2000		5000		SET_A (20,366)	
Mean filling	**DT**	**LGBM**	**DT**	**XGB**	**DT**	**XGB**
UtilityScore_mean	**0.150**	0.125	**0.091**	0.081	**0.088**	0.044
UtilityScore_std	0.117	0.070	0.021	0.050	0.016	0.009
F1Score_mean	0.106	0.125	0.094	0.105	0.098	0.076
F1Score_std	0.052	0.053	0.012	0.054	0.011	0.013
AUROC_mean	0.000	0.742	0.000	0.737	0.000	0.764
AUPRC_mean	0.000	0.089	0.000	0.081	0.000	0.090
Accuracy_mean	0.934	0.960	0.949	0.973	0.948	0.974
Accuracy_std	0.018	0.019	0.008	0.005	0.002	0.002
Baseline_mean	98.170	98.170	98.180	98.180	97.840	97.840
Total time (sec)	33.350	59.430	77.820	313.030	1313.90	4510.940

6.1.5 Forwards filling plus zero filling

The implementation of the forward filling method is based on already executed research related to the early prediction of sepsis. The forward filling method is used in two papers of the Top 5 submission papers of the PhysioNet Challenge. The inner workings of this method had already discussed in chapter 5.1.3. In short, this method fills in the missing data with existing data from an early point in time. Data points that are still missing after this operation will be filled with the value zero. The experiments conducted with this forward filling method for imputing the missing data will be discussed below. The results of the different used classifiers for the 400-1000 patient dataset experiments will all be shown. But for the larger datasets, only the two best performing classifiers, with the highest utility scores, are displayed.

First, experiments were conducted on the 400 patients dataset with the earlier mentioned classifiers. In Table 19, the results of these experiments are displayed. Here, there can be observed that the Decision Tree and Gradient Boosting classifier are the classifiers who obtain the two highest utility scores. For the results of the experiment on the 1000 patients dataset shown in Table 20, the same conclusion can be drawn as for the 400 patients experiment. Again, the Decision Tree and gradient boosting classifier obtain the highest utility score.

Table 19: Results of FFIL_0 experiment on the 400 patients dataset

400 patients	DT	XGB	ADA	GRAD	LGBM	LR
UtilityScore_mean	**0.240**	0.023	0.062	0.142	0.013	0.019
UtilityScore_std	0.297	0.065	0.106	0.257	0.088	0.040
F1Score_mean	0.155	0.067	0.060	0.093	0.047	0.036
F1Score_std	0.106	0.091	0.110	0.133	0.065	0.043
AUROC_mean	0.000	0.762	0.745	0.828	0.747	0.708
AUPRC_mean	0.000	0.157	0.153	0.209	0.166	0.081
Accuracy_mean	0.925	0.968	0.963	0.959	0.956	0.951
Accuracy_std	0.065	0.027	0.037	0.032	0.041	0.057
Baseline_mean	97.880	97.880	97.880	97.880	97.880	97.880
Total time (sec)	24.130	108.600	77.920	111.100	129.380	143.110

Table 20: Results of FFIL_0 experiment on the 1000 patients dataset

1000 patients	DT	XGB	ADA	GRAD	LGBM
UtilityScore_mean	0.073	0.058	0.085	**0.132**	0.090
UtilityScore_std	0.085	0.071	0.072	0.112	0.085
F1Score_mean	0.076	0.075	0.128	0.140	0.098
F1Score_std	0.050	0.087	0.118	0.099	0.074
AUROC_mean	0.000	0.794	0.773	0.782	0.795
AUPRC_mean	0.000	0.106	0.129	0.122	0.101
Accuracy_mean	0.962	0.974	0.979	0.966	0.966
Accuracy_std	0.018	0.010	0.008	0.017	0.018
Baseline_mean	98.260	98.260	98.260	98.260	98.260
Total time (sec)	195.300	212.630	219.080	264.950	253.310

The results of the forward filling experiments which were performed on the larger dataset are shown in Table 21. From these results, there can be noted that the Decision tree classifier achieves one of the highest utility scores throughout the different data size experiments.

For the experiments for 2000 and 5000 patients datasets, the Gradient Boosting and Light gradient boosting classifiers achieved the two best utility scores of all the used classifiers. Unfortunately, their utility score for the experiment performed on the training A dataset has deteriorated so much, compared to the previous experiments. A second item that can be deduced from the above table is that the performance of the XGBoost classifier increases as the dataset on which the experiment is performed gets larger. This results in that the XGBoost classifier achieves the second-highest utility score for the filling experiment on the training dataset A.

Table 21: Results of FFIL_0 experiment on the larger datasets

Dataset size	**2000**			**5000**			**Set_A**	
	DT	**GRAD**	**LGBM**	**DT**	**GRAD**	**LGBM**	**DT**	**XGB**
UtilityScore_mean	0.107	**0.232**	0.182	0.127	0.111	**0.150**	**0.050**	0.039
UtilityScore_std	0.113	0.097	0.088	0.072	0.077	0.067	0.029	0.037
F1Score_mean	0.092	0.180	0.154	0.107	0.119	0.137	0.070	0.059
F1Score_std	0.060	0.067	0.056	0.027	0.045	0.017	0.020	0.040
AUROC_mean	0.000	0.745	0.766	0.000	0.752	0.757	0.000	0.719
AUPRC_mean	0.000	0.125	0.108	0.000	0.081	0.078	0.000	0.069
Accuracy_mean	0.955	0.949	0.956	0.950	0.967	0.960	0.947	0.972
Accuracy_std	0.019	0.020	0.021	0.015	0.011	0.013	0.005	0.005
Baseline_mean	98.170	98.170	98.170	98.180	98.180	98.180	97.840	97.840
Total time (sec)	344.730	589.990	368.560	794.940	1410.450	890.830	1283.560	3336.840

6.1.6 Forwards filling plus mean filling

The method used for the set of experiments is similar to the previously discussed forward filling, method. The only difference is that after forwards filling, the remaining missing values will now be filled using the mean values of each feature (column). Similar to the previous experiments, first the experiments on the smaller datasets will be discussed followed by the experiment discussion on the "large" dataset. The results of the performed experiments for the forward filling followed by the mean filling method for the 400 and 1000 patients dataset are displayed in Table 22 and Table 23. For the experiments on the 400 patients dataset, there can visibly be observed that the Decision Tree classifier achieves the best util score (0.1534) by fare. The second best classifier is the AdaBoost classifier, which achieves a utility score of 0.0375. However, it has a higher accuracy score (0.9724) than the Decision Tree classifier (0.9595).

Table 22: Results of FFIL_mean experiment on the 400 patients dataset

400 patients	DT	XGB	ADA	GRAD	LGBM	LR
UtilityScore_mean	**0.153**	0.013	0.038	0.005	-0.003	0.014
UtilityScore_std	0.271	0.054	0.062	0.065	0.022	0.045
F1Score_mean	0.121	0.029	0.052	0.021	0.010	0.008
F1Score_std	0.114	0.069	0.093	0.064	0.022	0.025
AUROC_mean	0.000	0.740	0.701	0.816	0.726	0.652
AUPRC_mean	0.000	0.106	0.135	0.151	0.155	0.053
Accuracy_mean	0.960	0.972	0.972	0.971	0.965	0.967
Accuracy_std	0.024	0.020	0.019	0.015	0.037	0.034
Baseline_mean	97.880	97.880	97.880	97.880	97.880	97.880
Total time (sec)	78.080	103.420	81.260	116.190	111.160	151.430

The results of the experiments conducted on the 1000 patients dataset are more nuanced than those of the 400 patients dataset. The Decision Tree classifier still achieves the highest utility score but compared to the 400 patients experiment the utility score is significantly decreased. On the other hand, the AdaBoost classifier has higher performance scores, the utility score mean increases and the utility score std is decreased compared to the obtained score during experiments on the 400 patients dataset.

Table 23: Results of FFIL_mean experiments on the 1000 patients dataset

1000 patients	DT	XGB	ADA	GRAD	LGBM
UtilityScore_mean	**0.060**	0.005	0.042	0.003	0.002
UtilityScore_std	0.074	0.019	0.043	0.009	0.043
F1Score_mean	0.096	0.019	0.055	0.011	0.018
F1Score_std	0.081	0.035	0.053	0.014	0.055
AUROC_mean	0.000	0.763	0.760	0.778	0.775
AUPRC_mean	0.000	0.085	0.113	0.108	0.082
Accuracy_mean	0.976	0.980	0.978	0.980	0.978
Accuracy_std	0.007	0.006	0.008	0.005	0.008
Baseline_mean	98.260	98.260	98.260	98.260	98.260
Total time (sec)	173.820	228.400	208.900	259.110	192.260

In Table 24, the results of the experiment conducted on the large datasets are shown. In the first instance, the best performing classifiers of the previous experiments were used for the experiments on large datasets. However, only the Decision Tree classifier achieves in all the experiments one of the highest utility scores. Since the other best-performing classifiers of the previous experiment underperformed, the experiment was then executed for all the other, earlier mentioned, classifiers. For the experiments on the 2000 and 5000 patients dataset, the Decision Tree and Light Gradient boosting model achieve the highest utility score. However, it is not the case for the experiment performed on training dataset A. Here the performance of the LGBM classifier has drastically decreased (under 0.001%), but the XGBoost achieves the second "highest" utility (0.0016) score after the Decision Tree classifier. Based on the results of the experiment on training dataset A, this filling method does not perform well compared to the previously discussed filling methods.

Table 24: Results of the FFIL_mean experiments on the large datasets

	2000		5000		Set_A	
	DT	**LGBM**	**DT**	**LGBM**	**DT**	**XGB**
UtilityScore_mean	0.016	**0.029**	**0.029**	0.017	**0.034**	0.002
UtilityScore_std	0.043	0.045	0.026	0.019	0.049	0.003
F1Score_mean	0.037	0.040	0.060	0.033	0.058	0.004
F1Score_std	0.039	0.054	0.031	0.032	0.027	0.007
AUROC_mean	0.000	0.761	0.000	0.758	0.000	0.721
AUPRC_mean	0.000	0.080	0.000	0.073	0.000	0.079
Accuracy_mean	0.971	0.977	0.972	0.980	0.952	0.978
Accuracy_std	0.009	0.009	0.005	0.002	0.018	0.001
Baseline_mean	98.170	98.170	98.180	98.180	97.840	97.840
Total time (sec)	362.250	386.370	877.500	977.250	2991.040	6607.330

6.1.7 Filling experiment on the full dataset

All of the previously discussed filling methods will be experimented on with the full dataset. In Table 25, the results of the different filling methods experiments conducted on the full dataset are shown. For the experiments, it was opted to use both the Decision Tree and Gradient Boosting classifier for training a model. This decision was based on the experiments that were conducted using the "smaller" datasets combined with the additional augmented features. After the SIRS score was added and columns with high missing data were dropped, the overall obtained performance of the prediction increased. Firstly, from this table, we can derive that the Decision Tree classifier again achieves the highest utility score. The second thing that stands out is that when these results are compared to the results obtained on the training dataset A, it can be noted that the utility score mean has increased and at the same time, the utility score std has decreased. This change is also visible in the F1Score metric. It is also noticeable that the performance metrics improve as the dataset grows because it has a large collection of patient data, which will be used as a reference for a prediction.

Table 25: Results of the filling experiments on the full dataset

	FFIL_0		FFIL_mean		mean filling		zero filling	
Full dataset	**GB**	**DT**	**GB**	**DT**	**GB**	**DT**	**GB**	**DT**
UtilityScore_mean	0.002	**0.067**	0.001	**0.044**	0.007	**0.086**	0.007	**0.087**
UtilityScore_std	0.001	0.033	0.001	0.015	0.003	0.006	0.003	0.003
F1Score_mean	0.004	0.085	0.003	0.071	0.015	0.095	0.014	0.095
F1Score_std	0.002	0.026	0.002	0.016	0.006	0.005	0.007	0.003
AUROC_mean	0.783	0.000	0.785	0.000	0.798	0.000	0.800	0.000
AUPRC_mean	0.092	0.000	0.095	0.000	0.099	0.000	0.101	0.000
Accuracy_mean	0.982	0.961	0.982	0.971	0.982	0.954	0.982	0.954
Accuracy_std	0.001	0.007	0.001	0.002	0.001	0.002	0.001	0.002
Baseline_mean	98.200	98.200	98.200	98.200	98.200	98.200	98.200	98.200

6.2 Experiment to reliably predict the occurrence of sepsis

From the previous experiments, for data imputation, the zero filling and the forward filling plus zero filling are the two optimal methods. For model training, Decision Tree and Gradient Boosting are the two best candidates.

In this chapter, mainly four experiments are performed for their influence on the overall performances, namely dropping columns, and adding SIRS, SOFA, and sliding window attributes. The different data imputation methods and training models are applied in the experiment to compare their performances.

6.2.1 Dropping columns

From Table 7 in chapter 5.1, more than half of the columns are severely incomplete. For these columns, over 90% of the data is missing. Even after the data imputation of the forward filling followed by zero filling, the missing data is replaced either by its previous value or zero. Especially for the columns with a missing rate of over 98%, the useful information they provide is insignificant, much less than that provided by vital sign columns. Besides, the presence of the severely missing columns may interfere with the judgement of the algorithm. Since the values in them are either all zeros or all the same for one patient, the algorithm could be trained better without them. That is why the dropping columns experiments are performed.

With six different threshold values ranging namely 90%, 91%, 92%, 93%, 94% and 100%, six experiments were performed. Columns with a missing ratio larger than the threshold value were dropped from the dataset. Then, the dataset was imputed with forwarding filling plus the zero filling. After that, it was divided into five groups by the Patient_id for the K-Fold algorithm. For each iteration, a Decision Tree classifier was used to train the model by the training dataset. Evaluation metrics like utility scores, F1Score, AUROC, AUPRC and accuracy were performed on the test dataset. At last, for each evaluation metric, five results were obtained. Their means and standard derivations were calculated and represented in the following table.

Table 26: Results for dropping columns with different threshold values (K=5, Decision Tree, forward + zero filling)

Dropping columns with a missing ratio higher than	90%	91%	92%	93%	94%	100%
UtilityScore_mean	0.087	**0.101**	**0.100**	0.095	0.069	0.067
UtilityScore_std	0.021	0.030	0.030	0.038	0.024	0.033
F1Score_mean	0.082	0.088	0.092	0.092	0.081	0.085
F1Score_std	0.010	0.013	0.013	0.017	0.014	0.026
AUROC_mean	0.000	0.000	0.000	0.000	0.000	0.000
AUPRC_mean	0.000	0.000	0.000	0.000	0.000	0.000
Accuracy_mean	0.931	0.932	0.942	0.946	0.953	0.961
Accuracy_std	0.006	0.004	0.007	0.008	0.008	0.007
Baseline_mean	98.200	98.200	98.200	98.200	98.200	98.200

Besides Decision Tree, Gradient Boosting and Light Gradient Boosting are also good machine-learning models. With all the other procedures remaining the same, the above experiment was performed again with Gradient Boosting and Light Gradient Boosting Model (LGBM) in replacement of the Decision Tree machine-learning model. Below are the results.

Table 27: Results for dropping columns with different threshold values (K=5, Gradient Boosting, forward plus zero filling)

Dropping columns with a missing ratio higher than	90%	91%	92%	93%	94%	100%
UtilityScore_mean	0.002	0.003	0.003	0.003	0.002	0.001
UtilityScore_std	0.002	0.002	0.002	0.001	0.001	0.001
F1Score_mean	0.005	0.007	0.008	0.007	0.004	0.003
F1Score_std	0.004	0.005	0.004	0.001	0.002	0.002
AUROC_mean	0.777	0.775	0.779	0.779	0.781	0.785
AUPRC_mean	0.088	0.088	0.088	0.089	0.093	0.095
Accuracy_mean	0.982	0.982	0.982	0.982	0.982	0.982
Accuracy_std	0.001	0.001	0.001	0.001	0.001	0.001
Baseline_mean	98.200	98.200	98.200	98.200	98.200	98.200

Table 28: Results for dropping columns with different threshold values (K=5, Light Gradient Boosting, forward plus zero filling)

Dropping columns with a missing ratio higher than	90%	91%	92%	93%	94%	100%
UtilityScore_mean	0.018	0.014	0.028	0.011	0.011	0.014
UtilityScore_std	0.019	0.018	0.028	0.005	0.004	0.007
F1Score_mean	0.036	0.033	0.048	0.026	0.026	0.030
F1Score_std	0.024	0.025	0.033	0.009	0.007	0.013
AUROC_mean	0.766	0.766	0.773	0.771	0.776	0.780
AUPRC_mean	0.057	0.058	0.069	0.066	0.072	0.079
Accuracy_mean	0.977	0.977	0.977	0.979	0.980	0.980
Accuracy_std	0.003	0.001	0.003	0.001	0.002	0.000
Baseline_mean	98.200	98.200	98.200	98.200	98.200	98.200

From the results shown in Table 27 and Table 28, both Gradient Boosting and Light Gradient Boosting are not suitable here. They performed much worse than the Decision Tree. Although achieving an accuracy of 0.98, the UtilityScore_mean were either 0.001 or 0.01. That was not the result wanted in this experiment. Therefore, neither Gradient Boosting nor Light Gradient Boosting was used for the following experiments. Decision Tree was proved to be the best classifier and was used as the only training model for the experiments afterwards.

Besides the imputation method of the forward filling plus zero filling, the zero filling is also a good candidate to fill the missing values. It's just to fill all the missing values with zero, without forward filling before. It's performing better than the forward filling plus zero filling. How does it perform after some columns are dropped? With all the other procedures the same, the first experiment in 6.2.1 was performed again with only zero filling to fill the missing data. Below is the result.

Table 29: Results for dropping columns with different threshold values (K=5, Decision Tree, zero filling)

Dropping columns with a missing ratio higher than	90%	91%	92%	93%	94%	100%
UtilityScore_mean	0.078	0.075	0.081	0.081	0.086	0.087
UtilityScore_std	0.003	0.003	0.004	0.004	0.004	0.003
F1Score_mean	0.089	0.088	0.091	0.092	0.095	0.095
F1Score_std	0.003	0.002	0.002	0.002	0.004	0.003
AUROC_mean	0.000	0.000	0.000	0.000	0.000	0.000
AUPRC_mean	0.000	0.000	0.000	0.000	0.000	0.000
Accuracy_mean	0.078	0.075	0.081	0.081	0.086	0.087
Accuracy_std	0.003	0.003	0.004	0.004	0.004	0.003
Baseline_mean	0.089	0.088	0.091	0.092	0.095	0.095

Since the Decision Tree model does not provide the probability of the prediction result, both the AUROC and AUPRC are zeros for Table 26 and the results below. From Table 26, it's clear that the performance improves after dropping columns. Keeping all the columns, the UtilityScore_mean is only 0.067. After dropping some columns, the UtilityScore_mean can achieve as high as 0.101, much higher than before.

From Table 29, the performance worsened after dropping columns. The more columns are dropped, the lower the UtilityScore_mean is. Therefore, it's best to just keep all the columns for the zero filling test. But even with all the columns preserved, the best UtilityScore_mean it can achieve was only 0.087, while the forward filling plus zero filling can achieve 0.101. The goal is to design the algorithm that can achieve the highest utility score. Therefore, it is better to choose to drop columns with a missing ratio higher than 91% or 92% with the forward filling plus zero filling method.

In conclusion, after the comparisons of the different algorithm components, it's helpful to use forward Filing plus zero filling, Dropping columns and the Decision Tree training model from now on. The best UtilityScore_mean can be achieved currently was 0.101.

6.2.2 SIRS

As discussed in chapter 3.3.2.1, SIRS was once used in the definition of sepsis. It is calculated based on the four measurements: temperature, heart rate, respiratory rate and white blood cell. For the first three measurements, the missing rates are all very low. But for the last measurement, white blood cell (WBC), the missing rate was 93%. If drop the columns of more than 91% missing, the WBC column was also dropped. To calculate SIRS in this case, only temperature, heart rate, and respiratory rate can be used. It's named "partial SIRS".

In this part, two experiments were performed based on the above experiments. One was to add the partial SIRS column to the dataset which has been dropped the columns with more than 91% data missing. The other was to add the SIR column to the full dataset without dropping any column. All the other procedures are the same as those in chapter 6.2.1. The results are presented in the table below.

Table 30: Results for adding SIRS (K=5, Decision Tree)

	Drop columns > 91%		**All columns**	
	No SIRS	Adding SIRS (Partial SIRS)	No SIRS	Adding SIRS
UtilityScore_mean	0.101	0.096	0.067	0.085
UtilityScore_std	0.030	0.026	0.033	0.024
F1Score_mean	0.088	0.086	0.085	0.094
F1Score_std	0.013	0.011	0.026	0.014
AUROC_mean	0.000	0.000	0.000	0.000
AUPRC_mean	0.000	0.000	0.000	0.000
Accuracy_mean	0.932	0.932	0.961	0.956
Accuracy_std	0.004	0.005	0.007	0.007
Baseline_mean	98.200	98.200	98.200	98.200

As seen from the result, adding the partial SIRS score column to the "drop column > 91%" dataset was detrimental to the performance. The UtilityScore_mean decreased from 0.101 to 0.096. For the "All columns" dataset, although adding the SIRS score column increased the UtilityScore_mean, it is still much lower than the result of "drop column > 91%". Considering the utility score is the final object to optimize, the "drop column > 91%" method was chosen without adding the SIRS score column. In conclusion, adding the SIRS score is unhelpful for the following experiments.

6.2.3 SOFA Scores

The complete procedure to calculate the SOFA score has been discussed in chapter 3.3.2.2. Although the SOFA score is viewed as the best clinical score for the detection of sepsis, most measurements to calculate the SOFA score are severely missed in the dataset.

They are $\mathrm{SaO_2}, \mathrm{FiO_2}, \mathrm{MAP}, \mathrm{Bilirubin_{total}}, \mathrm{Platelets}, \mathrm{and\ Creatinine}$. (We should have used $\mathrm{PaO_2}$ for the calculation of the respiratory system SOFA score. But due to it is not present in the dataset, $\mathrm{SaO_2}$ is used instead). Except for the MAP, all the other measurements have a missing ratio of over 90%. It is impossible to add the SOFA score column with dropping columns. So here all the columns must be preserved for the data training.

In this experiment, all the columns were preserved and put in data training in the same way in chapter 6.2.1, SOFA score column is added to the training dataset.

Table 31: Results of adding SOFA attributes (K=5, Decision Tree)

	All columns	
	No SOFA	Adding SOFA
UtilityScore_mean	0.0674	0.0674
UtilityScore_std	0.0329	0.0329
F1Score_mean	0.0847	0.0847
F1Score_std	0.0255	0.0255
AUROC_mean	0.000	0.000
AUPRC_mean	0.000	0.000
Accuracy_mean	0.961	0.961
Accuracy_std	0.007	0.007
Baseline_mean	98.200	98.200

The result remained the same after adding the SOFA score column. Besides, the UtilityScore_mean is still very low compared to the utility score of 0.101 achieved by dropping columns. In conclusion, adding a SOFA score is not necessary.

6.2.4 Sliding Window Attributes

In this part, six different sliding window attributes for the vital signs are added to the dataset before the model training. They are:

1. Last hour: The measurement of the eight vital signs for the last hour
2. Min: The minimal measurement value for the last six hours
3. Max: The maximal measurement value for the last six hours
4. Minmaxdiff: The difference between Max and Min, calculated by Max minus Min
5. Mean: The mean value of the measurements for the last six hours
6. Std: The standard derivation of the measurements for the last six hours

All the sliding window attributes are obtained from the eight vital signs and calculated for each patient. If there are less than six measurement values before the current row, it will take all the available rows for the sliding window. For example, to calculate the Min for the measurement of a patient in the fourth hour, it will take the minimal value of the last three hours. For the first row of a patient, there are no measurements timely before. In this case, it will just take the vital signs of this line for the sliding window.

From the results above, currently, the best performance was achieved by only dropping columns with a missing ratio high than 91% or 92%. Based on the procedure discussed in 6.2.1, the six sliding window attributes are added one by one in sequence to see their influence on the utility score. If the UtilityScore_mean is increased after adding the attribute, it will be kept. If the UtilityScore_mean decreases, it will be dropped. If the UtilityScore_mean remains the same, changes of the other evaluation scores will be considered.

In this way, the best combination of the attributes can be found. It was first performed based on the "Drop columns > 91%" experiment. Below are the results.

Table 32: Results for adding sliding window attributes (Drop columns > 91%, K=5, Decision Tree)

Adding sliding window of	None	Last hour	Min	Min+ Max	Min+ Max+ Minmaxdiff	Min+ Max+ Minmaxdiff+ Mean	Min+ Max+ Minmaxdiff+ Std
UtilityScore_mean	0.101	0.078	0.110	0.113	0.120	0.107	**0.120**
UtilityScore_std	0.030	0.028	0.019	0.018	0.020	0.021	0.016
F1Score_mean	0.088	0.076	0.091	0.092	0.089	0.084	0.089
F1Score_std	0.013	0.012	0.008	0.008	0.007	0.009	0.006
AUROC_mean	0.000	0.000	0.000	0.000	0.000	0.000	0.000
AUPRC_mean	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Accuracy_mean	0.932	0.922	0.930	0.930	0.916	0.912	0.915
Accuracy_std	0.004	0.008	0.005	0.003	0.007	0.009	0.003
Baseline_mean	98.200	98.200	98.200	98.200	98.200	98.200	98.200

After the experiment, the UtilityScore_mean was increased from 0.101 to 0.120 with the addition of the Min, Max, Minmaxdiff, and Std attributes. The Last hour and mean attributes were showed to be harmful to the performance. The UtilityScore_mean dropped after adding them. Meanwhile, the Min, Max, and Minmaxdiff are showed to be helpful for the improvement of the UtilityScore_mean. It is worth mentioning that, after adding the Std attribute, although the UtilityScore_mean and F1Score_mean remained the same, both the UtilityScore_std and F1Score_std decreased. This shows that the Std attribute is beneficial for the improvement of the stability of the algorithm. Therefore, it is kept.

Adding sliding window attributes was only performed for the "Drop columns > 91%" test, and it should be also performed for the "Drop columns > 92%" test. Here the only two

combinations that gave the best UtilityScore_mean for the “Drop columns > 91%” test were tried. In Table 33, the result of this experiment is displayed.

Table 33: Results for adding sliding window attributes (Drop columns > 92%, K=5, Decision Tree)

Adding sliding window	None	Min+ Max+ Minmaxdiff	Min+ Max+ Minmaxdiff+ std
UtilityScore_mean	0.100	0.119	**0.127**
UtilityScore_std	0.030	0.029	0.026
F1Score_mean	0.092	0.093	0.096
F1Score_std	0.013	0.012	0.011
AUROC_mean	0.000	0.000	0.000
AUPRC_mean	0.000	0.000	0.000
Accuracy_mean	0.942	0.926	0.927
Accuracy_std	0.007	0.004	0.004
Baseline_mean	98.200	98.200	98.200

Therefore, the best UtilityScore_mean currently was 0.127. From the former results, the performance of the “Drop columns > 91%” and “Drop columns > 92%” tests were always nearly the same. However, the Std attribute showed a more positive influence for the “Drop columns > 92%” test. This is the best UtilityScore_mean that was achieved in this paper. Below is the result for each fold of that experiment that achieved the best result.

Table 34: Results for each fold of the best result

Fold	1	2	3	4	5
Utility Score	0.100	0.104	0.172	0.122	0.139
F1Score	0.085	0.088	0.113	0.092	0.104
Accuracy	0.925	0.929	0.925	0.921	0.932
AUROC	0.000	0.000	0.000	0.000	0.000
AUPRC	0.000	0.000	0.000	0.000	0.000
Baseline	98.200	98.200	98.200	98.200	98.200

In conclusion, adding Min, Max, Minmaxdiff, and Std attributes from the sliding window method is helpful for the algorithm.

7 DISCUSSION

7.1 Reflection on missing data imputation experiments

For the executed experiments, the first conclusion that can be taken is regarding the KNN filling missing data method. This filling method achieves the highest performance for the smaller data sets. Thus, it achieves a utility score of 0.2755 and 0.1523 respectively for the 400 and 1000 patients dataset. But for the large (2000, 5000 patients and training dataset A) datasets, it presents worse than the other filling methods discussed. An important observation is that the KNN filling method is time-intensive. This is the reason why the experiments were carried out until the 5000 patients dataset. Otherwise, it would have to take a very long time (multiple) to fill the missing data in the full dataset.

A second conclusion that can be drawn from missing data imputation experiments is that the Decision Tree classifier achieves very good performance metrics for the different sizes of datasets.

For imputing the missing data in the smaller (400-1000 patients) dataset sizes with the Decision Tree classifier, it is best to use either the mean or zero filling methods. These filling methods realize the following utility scores:

- Mean filling method: 400 patients: 0.1705 – 1000 patients: 0.1566
- Zero filling method: 400 patients: 0.1735 – 1000 patients: 0.1542

For the larger and more complete datasets, there can best be opted for the following configurations:

- Mean filling using Decision Tree
- Zero filling using Decision Tree
- Forward filling plus zero filling method using Gradient Boosting for 2000 patients and Light Gradient Boosting for the 5000 patients and Decision Tree for the full (training) dataset

These three filling methods respectively achieve good performance scores (utility score, F1-score, accuracy). But the mean and zero filling methods will give the highest utility score. Another observation that can be taken for the forward filling plus zero filling method is that the Gradient boosting classifier performs very poorly compared to the Decision Tree classifier on the full dataset.

To conclude the performed experiment for finding the most “optimal” filling method, the two best performing filling methods are:

- Mean filling method, which achieved a utility score of 0.086 on the full dataset
- Zero filling method, which achieved a utility score of 0.0871 on the full dataset

These data filling methods achieve overall the best performance scores for the different datasets. An honourable mentioning goes to the forwards filling method which is ranked as the third best-performing filling method.

In the top 5 papers of the PhysioNet Challenge, different values for K-fold were used. During the experiments, we also investigated what the influence would be if for example K=5 would be used for the K-fold operation. From these experiments, it was concluded that the algorithm achieves higher performance scores when K=5 instead of before where K =10. In Table 35, the performance differences between K= 5 and K= 10 are highlighted. Here, there can be noted that the filing of missing data experiment that was executed using 5-fold cross-validation achieves a higher utility score and F1Score. The time needed to perform the 5-fold experiment has significantly decreased from 160 seconds to 93 seconds.

Table 35: Result of experiment with different K-fold (Decision Tree)

Decision Tree-1000	10-Fold	5-Fold
UtilityScore_mean	0.073	**0.116**
UtilityScore_std	0.085	0.128
F1Score_mean	0.076	0.110
F1Score_std	0.050	0.055
AUROC_mean	0.000	0.000
AUPRC_mean	0.000	0.000
Accuracy_mean	0.962	0.951
Accuracy_std	0.018	0.011
Baseline_mean	98.260	98.260
Total time (sec)	159.270	93.220

The conclusion of the experiments performed for the different filling methods, the mean and zero filling methods are the most suitable for imputing the missing values in a dataset.

However, after adding extra 'new' features and omitting the columns with many missing data as discussed in chapter 5.2, these filling methods do not achieve significant performance improvements compared to the filling method experiments with only a filling method and model. On the other hand, the third-best performing filling method, the forward filling plus zero Fill method leads to an improvement in utility score performance.

The conclusions regarding the executed experiments for reliable prediction of sepsis will be discussed in the following chapter 7.2.

7.2 Reflection on reliably sepsis prediction

From the experiment results, it can be firstly concluded that dropping some columns with high missing ratios is of vital importance to achieving a high utility score. It increased the mean of the utility scores from 0.067 to 0.101. This was for the dataset filled with the forward filling plus zero filling. Highly missed columns here can promote the generation of erroneous results and interfere with the ability of the algorithm to make correct predictions. However, for the dataset filled with zero filling, dropping highly missed columns decreased the utility score.

This resulted from the difference in the data imputation method. The forward filling replaces many missing values with the valid values timely before it. For the columns with a missing ratio higher than 95%, it is filling the whole columns with only 5% valid data. For every patient, there might be no or only one valid measurement in an overall highly missed column. After forward filling, most of the values in this column become the same value. This is unhelpful to train the classification algorithm to predict sepsis and non-sepsis example. Thus, after dropping these columns, the utility score increased obviously.

However, for the zero filling, it is simply replacing all the missing values with zero. This works as keeping the missing values irrelevant for the model training and only considers the valid data during the training. Thus, the more columns preserved, the more valid information is preserved and the better the performance is. The best Utilityscore_mean zero filling method can achieve, 0.087, which is lower than 0.101. Hence, the forward filling plus zero filling and dropping columns with missing ratios over 91% or 92% were the best algorithm component currently.

The second conclusion is SIRS and SOFA score are unhelpful for the algorithm training. Both of them require the presence of some highly missed columns. As discussed above, these columns process less useful information and can interfere with the judgement of the algorithm. Meanwhile, both the SIRS and SOFA score are calculated based on these columns. Therefore, after adding the SIRS and SOFA attribute column, the utility score decreased.

The third conclusion is that the sliding window attributes are beneficial for the training. It is found that the addition of the min, max, difference of the min and max, and the standard derivation of the last six hours of the vital signs can improve the utility score. After adding all these attributes, the mean of the utility score increased from 0.101 to 0.127 for the dropping columns of the missing ratios higher than the 92% test group. Sliding window attributes are calculated based on the vital sign measurements, which have low missing ratios for most of the time. Due to the always presentence of the valid data, vital signs contain much useful information. With the sliding window, the information contained in the data ahead can be extracted and appended to the current training data.

This provides the training model with more useful information and improves the utility score further. The best mean of the utility score achieved in this paper is by adding the sliding window attributes, as shown in Table 34.

The inner working of the final algorithm is shown in Figure 7.1.

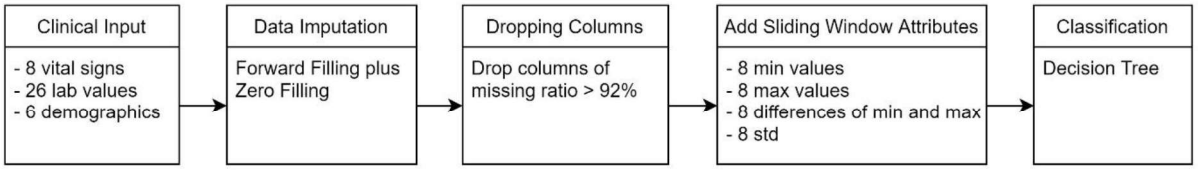

Figure 7.1: The block diagram of the algorithm

8 FUTURE WORK

Based on the results obtained and discussion of the experiments conducted, there are still some shortcomings in this thesis. Therefore, some possible improvements that can be implemented in the future for improving the achieved utility score will be mentioned.

Since this thesis is based on the PhysioNet Clinical Challenge, the participant's code is open-source available. Therefore, before the experimentation phase, research was conducted on the available "open source" code. However, at that time, we were only able to find the code of submission which achieved the highest utility score of the challenge. This was a good start, but his code is spread across multiple folders and files. To understand the code properly, we performed thorough research of the different code files and how they interact with each other. For this research, several days were spent, with the result that the implementation is very complex and required more time to fully understand the implemented (code) methodology. However, the useful parts that are used in this thesis, were described in the summary paper of their PhysioNet submission.

In addition to the code of the best submission, the code of other people who participated or did not participate in the PhysioNet Challenge was studied. Some of these sepsis-related projects made use of machine learning networks. However, for everything that is mentioned in this thesis, we need to be able to understand and explain the inner working of the algorithms. Therefore, it was decided not to implement/discuss these methods, since then the inner working of the algorithm needed to be discussed in the thesis. However, from the other examined sepsis prediction implementations, inspiration was taken for creating additional features and methods which were used to improve the utility score. One of the future works that could be performed is to examine the "new" available code projects which address the early prediction of sepsis. At the time of research, there weren't a lot of usable code projects but near the end of this thesis, the more useful code was found on GitHub.

A second item that can be researched in future works, is how the computational complexity can be improved. The developed code takes a long period to make a prediction. As mentioned in the introduction, each hour delay means an increase in mortality of 4-8%. Currently, a pipeline consisting of data splitting, dropping columns, feature augmentation, model training is being run to make a sepsis prediction. An experiment was performed on the full dataset (40,336 patients) with the following configuration:

- Forward filling plus zero filling
- Sliding window of Last + Minmax of past six hours
- 5-fold Cross-validation
- Drop columns with more than 91% of missing data

The next item that can be researched in the future, is the tuning of the classifier hyperparameters. Based on the existing work, it does not seem a bad idea to tune the hyperparameters of the used classifiers. For instance, some of the top five papers [21], [23] implemented the following hyperparameter tuning: limiting the depth of the choice trees, early stopping to avoid overfitting and defining a learning rate. Maybe these optimised classifiers will result in an improved utility score.

A final item that can be explored in future improvements is the use of deep learning to learn and train the data. Deep learning is more performant than the 'traditional' algorithms that were implemented in this thesis. Deep learning can achieve good results with limited knowledge of the data. In deep learning, larger labelled datasets are used together with neural networks which learn features from the dataset without the need for manual extraction of the features [3].

Currently, we have already taken a head start for optimizing the utility score. For improving the utility score, we performed experiments with two approaches. The first method is to balance the unbalanced dataset by random oversampling of sepsis patients. The second method investigated is to use an ensemble of a classifier. Both methods were performed on training dataset A and B. The highest utility score obtained for the oversampling method is 0.350 and when using an ensemble of classifiers a utility score of 0.3803 was obtained. Unfortunately, it has not been possible so far to run these two methods on the full dataset.

9 CONCLUSION

The purpose of this chapter was to use machine learning to build an algorithm to predict sepsis from clinical data. The target was to get the highest utility score possible. To accomplish this, data imputation was first carried out, followed by various dataset procedures to train the algorithm. To ensure that each sample was used at least once for the test dataset, a K-Fold of five was used. For data imputation and model training, there were numerous options. Only those who scored highest on utility were selected. The forward filling plus zero filling method and the Decision Tree model were shown to be the best candidates after all the experiments.

In addition, other methods were used to increase improve the performance even further. Dropping columns of missing ratios greater than 92%, adding the min, max, gap between min and max and the standard derivation of one patient's last six hours vital signs were found to be beneficial. They were all included in the final algorithm which performed the best among all. The best performance was represented by the five utility scores with a mean of 0.127 and a standard derivation of 0.026.

The best utility scores achieved in this thesis was low compared to those in the top 5papers for the PhysioNet Challenge 2019. The time it took to train the algorithm was also very long. In the future, the code from other participants for the Challenge will be learned and their methods will be applied to increase the utility score further. The complexity of the algorithm will also be optimized to accelerate the training process of the algorithm.

10 BIBLIOGRAPHY

[1] 'The effect of age on the development and outcome of adult sepsis - PubMed'. https://pubmed.ncbi.nlm.nih.gov/16374151/ (accessed Jan. 03, 2021).

[2] P. E. Marik, 'Early Management of Severe Sepsis: Concepts and Controversies', *CHEST*, vol. 145, no. 6, pp. 1407–1418, Jun. 2014, doi: 10.1378/chest.13-2104.

[3] J. Soong and N. Soni, 'Sepsis: recognition and treatment', *Clin Med (Lond)*, vol. 12, no. 3, pp. 276–280, Jun. 2012, doi: 10.7861/clinmedicine.12-3-276.

[4] European Sepsis Alliance, 'Sepsis — European Sepsis Alliance', *Sepsis — European Sepsis Alliance*. https://www.europeansepsisalliance.org/sepsis (accessed Dec. 03, 2020).

[5] 'World Sepsis Day - Clinical Excellence Commission'. https://www.cec.health.nsw.gov.au/keep-patients-safe/Deteriorating-patient-program/Sepsis/world-sepsis-day (accessed Dec. 11, 2020).

[6] C. W. Seymour *et al.*, 'Time to Treatment and Mortality during Mandated Emergency Care for Sepsis', *N Engl J Med*, vol. 376, no. 23, pp. 2235–2244, Jun. 2017, doi: 10.1056/NEJMoa1703058.

[7] M. A. Reyna *et al.*, 'Early Prediction of Sepsis from Clinical Data: the PhysioNet/Computing in Cardiology Challenge 2019', in *2019 Computing in Cardiology (CinC)*, Sep. 2019, p. Page 1-Page 4. doi: 10.23919/CinC49843.2019.9005736.

[8] 'What is Sepsis or Septicemia (Blood Infection)?', *WebMD*. https://www.webmd.com/a-to-z-guides/sepsis-septicemia-blood-infection (accessed Apr. 25, 2021).

[9] 'Sepsis', *Wikipedia*. Apr. 04, 2021. Accessed: Apr. 15, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sepsis&oldid=1015858499

[10] 'Sepsis: Symptoms, Causes, Treatment, Risks & More', *Healthline*, Aug. 31, 2018. https://www.healthline.com/health/sepsis (accessed Nov. 20, 2020).

[11] G. en wetenschap, 'Bloedvergiftiging (sepsis) · Gezondheid en wetenschap', *gezondheidenwetenschap.be*, Jan. 15, 2020. https://www.gezondheidenwetenschap.be/richtlijnen/bloedvergiftiging-sepsis (accessed Nov. 20, 2020).

[12] A. Kumar *et al.*, 'Duration of hypotension before initiation of effective antimicrobial therapy is the critical determinant of survival in human septic shock*':, *Critical Care Medicine*, vol. 34, no. 6, pp. 1589–1596, Jun. 2006, doi: 10.1097/01.CCM.0000217961.75225.E9.

[13] CDC, 'Sepsis is a medical emergency. Time matters.', *Centers for Disease Control and Prevention*, Aug. 27, 2020. https://www.cdc.gov/sepsis/what-is-sepsis.html (accessed Nov. 25, 2020).

[14] C. W. Seymour *et al.*, 'Assessment of Clinical Criteria for Sepsis: For the Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)', *JAMA*, vol. 315, no. 8, p. 762, Feb. 2016, doi: 10.1001/jama.2016.0288.

[15] M. Singer *et al.*, 'The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)', *JAMA*, vol. 315, no. 8, p. 801, Feb. 2016, doi: 10.1001/jama.2016.0287.

[16] K.-M. Kaukonen, M. Bailey, S. Suzuki, D. Pilcher, and R. Bellomo, 'Mortality Related to Severe Sepsis and Septic Shock Among Critically Ill Patients in Australia and New Zealand, 2000-2012', *JAMA*, vol. 311, no. 13, p. 1308, Apr. 2014, doi: 10.1001/jama.2014.2637.

[17] G. S. Martin, D. M. Mannino, S. Eaton, and M. Moss, 'The Epidemiology of Sepsis in the United States from 1979 through 2000', *N Engl J Med*, vol. 348, no. 16, pp. 1546–1554, Apr. 2003, doi: 10.1056/NEJMoa022139.

[18] M. Shankar-Hari *et al.*, 'Developing a New Definition and Assessing New Clinical Criteria for Septic Shock: For the Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)', *JAMA*, vol. 315, no. 8, p. 775, Feb. 2016, doi: 10.1001/jama.2016.0289.

[19] I. Cohen, 'Time series-Introduction', *Time series-Introduction. A time series is a series of data… | by Idit Cohen | Towards Data Science*, Oct. 04, 2019. https://towardsdatascience.com/time-series-introduction-7484bc25739a (accessed Apr. 14, 2021).

[20] J. Morrill, A. Kormilitzin, A. Nevado-Holgado, S. Swaminathan, S. Howison, and T. Lyons, 'The Signature-Based Model for Early Detection of Sepsis from Electronic Health Records in the Intensive Care Unit', presented at the 2019 Computing in Cardiology Conference, Dec. 2019. doi: 10.22489/CinC.2019.014.

[21] J. Anda Du, N. Sadr, and P. de Chazal, 'Automated Prediction of Sepsis Onset Using Gradient Boosted Decision Trees', presented at the 2019 Computing in Cardiology Conference, Dec. 2019. doi: 10.22489/CinC.2019.423.

[22] M. Zabihi, S. Kiranyaz, and M. Gabbouj, 'Sepsis Prediction in Intensive Care Unit Using Ensemble of XGboost Models', presented at the 2019 Computing in Cardiology Conference, Dec. 2019. doi: 10.22489/CinC.2019.238.

[23] X. Li, Y. Kang, X. Jia, J. Wang, and G. Xie, 'TASP: A Time-Phased Model for Sepsis Prediction', presented at the 2019 Computing in Cardiology Conference, Dec. 2019. doi: 10.22489/CinC.2019.049.

[24] J. Singh, K. Oshiro, R. Krishnan, M. Sato, T. Ohkuma, and N. Kato, 'Utilizing Informative Missingness for Early Prediction of Sepsis', presented at the 2019 Computing in Cardiology Conference, Dec. 2019. doi: 10.22489/CinC.2019.280.

[25] M. Poeze, G. Ramsay, H. Gerlach, F. Rubulotta, and M. Levy, 'An international sepsis survey: a study of doctors' knowledge and perception about sepsis', *Crit Care*, vol. 8, no. 6, pp. R409-413, Dec. 2004, doi: 10.1186/cc2959.

[26] R. Daniels, 'Surviving the first hours in sepsis: getting the basics right (an intensivist's perspective)', *Journal of Antimicrobial Chemotherapy*, vol. 66, no. Supplement 2, pp. ii11–ii23, Apr. 2011, doi: 10.1093/jac/dkq515.

[27] E. Rivers *et al.*, 'Early Goal-Directed Therapy in the Treatment of Severe Sepsis and Septic Shock', *New England Journal of Medicine*, vol. 345, no. 19, pp. 1368–1377, Nov. 2001, doi: 10.1056/NEJMoa010307.

[28] Q. Mao *et al.*, 'Multicentre validation of a sepsis prediction algorithm using only vital sign data in the emergency department, general ward and ICU', *BMJ Open*, vol. 8, no. 1, p. e017833, Jan. 2018, doi: 10.1136/bmjopen-2017-017833.

[29] H. Burdick *et al.*, 'Validation of a machine learning algorithm for early severe sepsis prediction: a retrospective study predicting severe sepsis up to 48 h in advance using a diverse dataset from 461 US hospitals', *BMC Medical Informatics and Decision Making*, vol. 20, no. 1, p. 276, Oct. 2020, doi: 10.1186/s12911-020-01284-x.

[30] A. Vellido, V. Ribas, C. Morales, A. Ruiz Sanmartín, and J. C. Ruiz Rodríguez, 'Machine learning in critical care: state-of-the-art and a sepsis case study', *BioMedical Engineering OnLine*, vol. 17, no. 1, p. 135, Nov. 2018, doi: 10.1186/s12938-018-0569-2.

[31] M. Komorowski, L. A. Celi, O. Badawi, A. C. Gordon, and A. A. Faisal, 'The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care', *Nature Medicine*, vol. 24, no. 11, Art. no. 11, Nov. 2018, doi: 10.1038/s41591-018-0213-5.

[32] J. Garnacho-Montero and I. Martín-Loeches, 'Clinical management of sepsis can be improved by artificial intelligence: no', *Intensive Care Med*, vol. 46, no. 2, pp. 378–380, Feb. 2020, doi: 10.1007/s00134-020-05947-1.

[33] 'Systemic inflammatory response syndrome', *Wikipedia*. Mar. 13, 2021. Accessed: Apr. 20, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Systemic_inflammatory_response_syndrome&oldid=1011981384

[34] R. C. Bone *et al.*, 'Definitions for sepsis and organ failure and guidelines for the use of innovative therapies in sepsis. The ACCP/SCCM Consensus Conference Committee. American College of Chest Physicians/Society of Critical Care Medicine', *Chest*, vol. 101, no. 6, pp. 1644–1655, Jun. 1992, doi: 10.1378/chest.101.6.1644.

[35] P. E. Marik and A. M. Taeb, 'SIRS, qSOFA and new sepsis definition', *J Thorac Dis*, vol. 9, no. 4, pp. 943–945, Apr. 2017, doi: 10.21037/jtd.2017.03.125.

[36] J.-L. Vincent *et al.*, 'The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure', *Intensive Care Med*, vol. 22, no. 7, pp. 707–710, Jul. 1996, doi: 10.1007/BF01709751.

[37] 'SOFA score', *Wikipedia*. Dec. 27, 2020. Accessed: Apr. 13, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=SOFA_score&oldid=996613886

[38] 'Serial Evaluation of the SOFA Score to Predict Outcome in Critically Ill Patients | Critical Care Medicine | JAMA | JAMA Network'. https://jamanetwork.com/journals/jama/fullarticle/194262 (accessed Dec. 21, 2020).

[39] 'What is Sepsis', *Sepsis Alliance*. https://www.sepsis.org/sepsis-basics/what-is-sepsis/ (accessed Nov. 20, 2020).

[40] E. P. Raith *et al.*, 'Prognostic Accuracy of the SOFA Score, SIRS Criteria, and qSOFA Score for In-Hospital Mortality Among Adults With Suspected Infection Admitted to the Intensive Care Unit', *JAMA*, vol. 317, no. 3, pp. 290–300, Jan. 2017, doi: 10.1001/jama.2016.20328.

[41] Y. Freund *et al.*, 'Prognostic Accuracy of Sepsis-3 Criteria for In-Hospital Mortality Among Patients With Suspected Infection Presenting to the Emergency Department', *JAMA*, vol. 317, no. 3, pp. 301–308, Jan. 2017, doi: 10.1001/jama.2016.20329.

[42] 'GPU Accelerated Computing with Python', *NVIDIA Developer*, Nov. 19, 2013. https://developer.nvidia.com/how-to-cuda-python (accessed Apr. 20, 2021).

[43] 'Dask documentation - Scheduling', *Dask documentation*. scheduling.html (accessed Apr. 20, 2021).

[44] 'Machine Learning - Home', *Coursera*. https://www.coursera.org/learn/machine-learning/home/week/9 (accessed Apr. 20, 2021).

[45] C. Sehra, 'Decision Trees Explained Easily', *Medium*, Nov. 30, 2020. https://chiragsehra42.medium.com/decision-trees-explained-easily-28f23241248 (accessed Apr. 22, 2021).

[46] 'sklearn.tree.DecisionTreeClassifier — scikit-learn 0.24.1 documentation'. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (accessed Apr. 22, 2021).

[47] V. Morde, 'XGBoost Algorithm: Long May She Reign!', *Medium*, Apr. 08, 2019. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d (accessed Apr. 22, 2021).

[48] 'What is XGBOOST? | Data Science and Machine Learning', *Kaggle*. https://www.kaggle.com/getting-started/145362 (accessed Apr. 22, 2021).

[49] G. L. Team, 'AdaBoost Algorithm: Boosting Algorithm in Machine Learning', *GreatLearning Blog: Free Resources what Matters to shape your Career!*, May 28, 2020. https://www.mygreatlearning.com/blog/adaboost-algorithm/ (accessed Apr. 22, 2021).

[50] ‘AdaBoost Classifier in Python’, *DataCamp Community*, Nov. 20, 2018. https://www.datacamp.com/community/tutorials/adaboost-classifier-python (accessed Apr. 22, 2021).

[51] ‘sklearn.ensemble.AdaBoostClassifier — scikit-learn 0.24.1 documentation’. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html (accessed Apr. 22, 2021).

[52] ‘Understanding XGBoost Algorithm | What is XGBoost Algorithm?’ https://www.mygreatlearning.com/blog/xgboost-algorithm/ (accessed Apr. 22, 2021).

[53] ‘A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning’. https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ (accessed May 03, 2021).

[54] ‘Features — LightGBM 3.2.1.99 documentation’. https://lightgbm.readthedocs.io/en/latest/Features.html (accessed Apr. 22, 2021).

FACULTY OF ENGINEERING TECHNOLOGY
GROUP T LEUVEN CAMPUS
Andreas Vesaliusstraat 13
3000 LEUVEN, België
tel. + 32 16 30 10 30
fet.groupt@kuleuven.be
www.fet.kuleuven.be