LUT University
School of Engineering Science
Industrial Engineering and Management
Master's Thesis

Einari Junter

PREDICTING SEPSIS IN THE INTENSIVE CARE UNIT USING MACHINE LEARNING

Examiners: Professor Pasi Luukka
PhD Christoph Lohrmann

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Degree Programme in Industrial Engineering and Management

Einari Junter

Predicting sepsis in the Intensive Care Unit using Machine Learning

Master's thesis
2020
79 pages, 25 figures and 22 tables
Examiners: Professor Pasi Luukka and PhD Christoph Lohrmann

Keywords: sepsis, machine learning, intensive care unit, health care, MIMIC-III, sepsis-3

Sepsis is a major burden to modern hospitals in terms of cost and death. Sepsis is a condition that lacks a diagnostic test making it hard to detect timely even for experienced medical professionals. The objective of this thesis is to examine different machine learning models and their performance to help in predicting patients in risk of sepsis.

The topic of this thesis is predicting sepsis in the intensive care unit using machine learning. Relevant literature of the medical and machine learning background was examined. Related work studying predicting sepsis with machine learning was observed.

This thesis uses the MIMIC-III dataset and the currently latest definition of sepsis, Sepsis-3. Heart rate, respiratory rate, systolic and diastolic blood pressures, SpO2, temperature, bilirubin, creatinine, glucose, lactate, age and gender are used to make the predictions. A six-hour data window was used.

Models of four different types were developed during this thesis project. A random forest model, two different gradient tree boosting models and a LSTM deep learning model were trained and tested. This thesis also compares the clinical setting performance of a model using all the data available from the patients' intensive care unit stays during the training compared to a model that uses only a fixed window close to the onset of sepsis.

The study cohort consisted of 6218 ICU stays. The patient-wise prevalence of sepsis in the study cohort was 9.2%. The best performing model, XGBoost, achieved 0.924 AUROC and 0.700 AUPRC at onset, 0.860 AUROC and 0.431 AUPRC at six hours before onset.

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tuotantotalouden koulutusohjelma

Einari Junter

Sepsiksen ennustaminen teho-osastolla koneoppimisen avulla

Diplomityö
2020
79 sivua, 25 kuvaa ja 22 taulukkoa
Tarkastajat: Professori Pasi Luukka ja TkT Christoph Lohrmann

Hakusanat: sepsis, koneoppiminen, teho-osasto, terveydenhuolto, MIMIC-III, sepsis-3

Sepsis on suuri taakka modernille terveydenhuololle kustannuksissa ja potilaskuolemissa mitattuna. Diagnostisen testin puuttuminen tekee sepsiksestä vaikean tunnistaa potilaassa kokeneellekin terveydenhuollon ammattilaiselle. Tämän diplomityön tavoite on tutkia eri koneoppimismalleja ja niiden suorituskykyä ennustamaan teho-osaston potilaat, joilla on riski sepsikseen.

Diplomityön aihe on sepsiksen ennustaminen teho-osastolla koneoppimisen avulla. Terveydenhuollon ja koneoppimiseen liittyvää kirjallisuutta on tutkittu työtä varten. Aiemmin toteutettuja tutkimuksia sepsiksen ennustamisesta koneoppimisen avulla on käyty läpi.

Tämä työ käyttää datana MIMIC-III potilasdataa. Sepsiksen määritelmänä käytössä on työn kirjoitushetkellä viimeisin määritelmä, Sepsis-3. Potilaan syke, hengitystiheys, systolinen ja diastolinen verenpaine, SpO2, lämpötila, bilirubiini, kreatiniini, glukoosi, laktaatti, ikä ja sukupuoli ovat käytössä ennustusten tekemiseen. Kuuden tunnin dataikkunaa käytettiin.

Neljä erilaista mallia kehitettiin työn aikana. Random forest, kaksi eri gradient tree boosting mallia, sekä LSTM-syväoppimismalli opetettiin ja testattiin. Työ tarkastelee myös kliinisen asetelman suorituskykyä vertaillen kahta XGBoost mallia, jotka kehitettiin potilasdataa eri tavoin hyödyntäen. Ensimmäinen vertailun malleista kehitettiin käyttäen dataa potilaiden koko teho-osastokäynnin ajalta, kun taas toinen käyttäen vain yksittäistä dataikkunaa jokaiselle potilaalle.

Tutkimuskohortti sisälsi 6218 teho-osastokäyntiä. Potilaskohtainen sepsiksen prevalenssi kohortissa oli 9.2%. Parhaiten suoriutunut malli, XGBoost, saavutti 0.924 AUROC- ja 0.700 AUPRC-tuloksen kun ennustus tehtiin juuri ennen sepsiksen alkua, sekä 0.860 AUROC- ja 0.431 AUPRC-tuloksen kun ennustus tehtiin kuusi tuntia ennen sepsiksen ilmenemistä.

PREFACE

I want to thank GE Healthcare and Chief Scientist Hanna Viertiö-Oja for trusting me with thesis opportunity. The journey has taught me a great deal, and the work that the GE Healthcare team does to improve lives of hospitalized people keeps on inspiring me.

Special thanks go to my advisor Hanna Saarinen for guiding me forward during this thesis work. The discussions and feedback from you were interesting, helpful and made it possible for me to achieve the goals we had set. Thank you for all the team members for your support and ideas. Additionally, a big thanks to my supervisor Professor Pasi Luukka for the great comments and feedback.

Lastly, I would like to thank my friends, family and my girlfriend Hanna for supporting me during the university studies and this thesis. I truly appreciate the encouragements and unconditional help that pushed me forward during both. Your presence was even more important during the disruptive times caused by the COVID-19.

Espoo, 8.5.2020
Einari Junter

CONTENTS

LIST OF TABLES

LIST OF FIGURES

colored crosses represent aggregated values of the previous hour. Slight jitter has been applied to the markers in the lower plot to make such values visible that have equal values between two aggregations. The plots share the x-axis.

Figure 16 – Cross-validation results for the random forest model. Each subplot represents the mean AUROC as the function of a hyperparameter.

Figure 17 – Cross-validation results for the gradient boosting model. Each subplot represents the mean AUROC as the function of a hyperparameter.

Figure 18 – Cross-validation results for the XGBoost model. Each subplot represents the mean AUROC as the function of a hyperparameter.

Figure 19 – Cross-validation results for the LSTM model. Each subplot represents the mean AUROC as the function of a hyperparameter.

Figure 20 – Illustration of the different prediction times. The blue bar represents the data window. The dashed line represents the gap. The box with filled line pattern is the prediction window. The orange dot represents the onset time.

Figure 21 – Model performance at onset. The left plot shows the Receiver Operating Characteristic curves and the right plot the Precision-Recall curves of the models.

Figure 22 – The models' performance as a function of the prediction gap length. The left plot shows the AUROCs and the right plot the AUPRCs of the models.

Figure 23 – An illustration of hourly prediction for a septic patient during their whole ICU stay. The blue data window and the black prediction window are slid across the whole length of the stay to achieve hourly predictions.

Figure 24 – Datapoint-wise performance of the continuous and fixed models on the test set for hourly predictions during the whole ICU stays.

Figure 25 – Patient-wise performance of the continuous and fixed models on the test set for hourly predictions during the whole ICU stays.

ABBREVIATIONS

AUC	Area Under the Curve
AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the Receiver Operating Characteristic
CI	Confidence Interval
EHR	Electronic Health Record
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GB	Gradient Boosting
ICU	Intensive Care Unit
LOS	Length of Stay
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
NN	Neural Network
NPV	Negative Predictive Value
PPV	Positive Predictive Value
PR	Precision-Recall
qSOFA	quick Sequential Organ Failure Assessment
RF	Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SD	Standard Deviation
SI	Suspected Infection
SIRS	Systemic Inflammatory Response Syndrome
SOFA	Sequential Organ Failure Assessment
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate

1 INTRODUCTION

Sepsis is responsible for over 50 % of deaths and over 24 billion dollar costs in the United States hospitals, all the while only accounting for 3,6 % of hospital stays (Paoli *et al.*, 2018). This makes it one of the highest burdens to the modern health care system. Due to the yet uncertain pathobiology of sepsis and a lack of a gold standard diagnostic test for it (Weis *et al.*, 2017), the possibilities of Machine Learning (ML) tools to capture patterns from patients' clinical signs are assumed to be highly valuable in assisting medical personnel to detect sepsis early.

The recent increase of monitoring and available Electronic Health Record (EHR) data within hospitals, especially in the Intensive Care Unit (ICU), now enables using a data science approach to build ML-based detection tools. ICU is also a relevant data source for septic patients, as most of them require intensive care actions through their treatment. The present practice to detect sepsis heavily relies on the clinician's ability to detect sometimes subtle changes in the patients' condition, or on severity score-based alarms. The problem with the first approach is that clinicians may be unable to detect changes until the patient's condition has become too unstable. In contrast, the threshold-based alarms cause many false positives leading to alarm fatigue.

The objective of this thesis work is to identify and study ML methods for predicting sepsis in the ICU and compare their performance for the MIMIC-III dataset (Johnson *et al.*, 2016). The reason behind the objective is the algorithm's potential to predict sepsis earlier and with higher certainty, which can lead to lives saved and improved care experience for both the care givers and takers. The main research question of this thesis is:

- What methods can be used to effectively predict sepsis during the ICU stay?

The following sub-questions help to break the main research question into smaller parts:

- What machine learning methods have been used in related studies to predict sepsis?
- How do these methods perform on the MIMIC-III dataset?

- What approaches have been used for extracting sepsis onset labels from the EHR data?

The structure of this thesis is as follows. Chapter 2 and Chapter 3 present the background of medical topics and machine learning, respectively. Chapter 4 describes related work that has supported the selection of suitable variables, pre-processing steps and machine learning algorithms for this thesis. Chapter 5 and Chapter 6 describe the dataset and methods used for building the predictive model, respectively. The experiments performed and the results obtained from them are presented in Chapter 7. Finally, Chapter 8 ends this thesis in discussion and conclusions.

2 MEDICAL BACKGROUND

This chapters presents the medical background related to the thesis. First, the concept of an Intensive Care Unit is presented. Further, the collection of Electronic Health Records is discussed. Lastly, the chapter ends with sepsis and its definitions.

2.1 Intensive Care Unit

The Intensive Care Unit (ICU) is a special hospital department that usually contains the patients with most critical conditions needing more intensive care than the rest. Since the injuries or illnesses present in ICU patients are life-threatening, they also require more intensive monitoring. To perform the monitoring, the ICUs are equipped with a wider range of different high-end patient-monitoring devices compared to the rest of the hospital. They also have more staff per patient, that gather more information about the patients' vital signs and laboratory tests that aim to describe the state of the patients (Ayers *et al.*, 2007). The availability of monitored information in the ICU makes it the preferred target for machine learning tasks, as the predictive algorithms greatly improve with more available data (Pollard and Celi, 2017).

Typical diseases treated in the ICU are major traumas, severe burns, respiratory failures, organ transplants and complex surgeries. The patients are often in need of artificial ventilation, dialysis or drugs to raise dropped blood pressure (Ayers *et al.*, 2007). There exist a variety of different variants of ICUs that specialize in treating specific medical requirements or patients, but this thesis considers an ICU from a more general setting.

2.2 Electronic Health Records

Electronic Health Records (EHR) are systems of electronically stored collections of patients' health information in digital format. They represent longitudinal data that are routinely collected in health care. They generally contain information about patients' demographic, vital statistics, administrative, claims, clinical, and patient-centred data. The EHR systems vary by geographic location and can be designed differently depending on a country's hospital culture. Systems designed primarily for billing can struggle in areas of clinical work support and vice

versa. The EHRs can also contain data from only a single hospital unit (like ICU), include data from a wider group of hospital-wide units or even link data across larger inter-hospital systems. EHRs have gained popularity due to improved quality of healthcare and its ability to capture billing data of hospital care. (Cowie *et al.*, 2017)

Records in EHRs provide a possibility for researchers to perform an analysis on patient care after the care period. The EHR data presents an opportunity to seek knowledge from the data to steer towards the direction of individualized patient care. The ideal EHR system would store all possible care and patient related data from as many patients and hospitals as possible. The current limitations in EHR systems available on the market often mean that some part of information that is available to the medical care professionals at the time of care is often lost. Missing data in the dataset derived from EHR theoretically mean a suboptimal performance of the ML algorithms developed using it. (MIT Critical Data, 2016)

2.3 Sequential Organ Failure Assessment Score

The Sequential Organ Failure Assessment score (SOFA) is a scoring system used to evaluate patients' health and status during their stay in the ICU (Weis *et al.*, 2017). Multiple Organ Failure is a major cause of morbidity and mortality in critically ill patients. The SOFA score was developed as a result of a consensus meeting of working group on sepsis-related problems of the European Society of Intensive Care Medicine in 1994 to create a sepsis-related organ failure assessment (SOFA) score. The SOFA score consists of six simple objective variables, that are routinely collected from patients in the ICU and cover the main functions and organs of a body. The score's six variables cover the respiratory, cardiovascular, hepatic, coagulation, renal and neurological systems. Each variable is chosen so that its calculated value best represents the state of an organ system (Vincent *et al.*, 1996). The variables used in the SOFA score are:

- PaO2/FiO2 or mechanical ventilation required (PaO2 = partial pressure of oxygen, FiO2 = fraction of inspired oxygen)
- Glasgow Coma Scale
- Mean Arterial Pressure (MAP), or administration of vasopressors required
- Bilirubin
- Platelets
- Creatinine, or urine output

Each variable is scored with a value from 0 to 4, based on the severity. The scores of each variable are added up and the sum represents the total SOFA score of the patient. The lower the SOFA score is, the less severe the state of sequential organ failures is rated. The variables for subcomponents of the score are represented in Table 1. (Weis *et al.*, 2017)

Table 1 – Sequential Organ Failure Assesment score (Weis *et al.*, 2017)**.**

	Score				
System	**0**	**1**	**2**	**3**	**4**
Respiration					
PaO_2/FiO_2, mm Hg (kPa)	≥ 400 (53.3)	< 400 (53.3)	< 300 (40)	< 200 (26.7) with respiratory support	< 100 (13.3) with respiratory support
Coagulation					
Platelets, $\times 10^3/\mu L$	≥ 150	< 150	< 100	< 50	< 20
Liver					
Bilirubin, mg/dL (µmol/L)	< 1.2 (20)	1.2-1.9 (20-32)	2.0-5.9 (33-101)	6.0-11.9 (102-104)	
Cardiovascular	MAP ≥ 70 mm Hg	MAP < 70 mm Hg	Dopamine < 5 or dobutamine (any dose)[1]	Dopamine 5.1-15 or epinephrine ≤ 0.1 or norepinephrine ≤ 0.1[1]	Dopamine > 15 or epinephrine > 0.1 or norepinephrine > 0.1[1]
Central nervous system					
Glasgow Coma Scale score[2]	15	13-14	10-12	6-9	< 6
Renal					
Creatinine, mg/dL (µmol/L)	< 1.2 (20)	1.2-1.9 (110-170)	2.0-3.4 (171-299)	3.5-4.9 (300-440)	> 5.0 (440)
Urine output, mL/d				< 500	< 200

Abbreviations: FiO_2, fraction inspired oxygen; MAP, mean arterial pressure; PaO_2, partial pressure of oxygen.

[1] Catecholamine doses are given as µg/kg/min for at least 1 hour.

[2] Glasgow Coma Scale scores range from 3-15; higher score indicates better neurological function

2.4 Sepsis

Sepsis is a life-threatening condition where the body's response to an infection causes injury to its own organs and tissues. A bacterial infection is most common, but it can also be viral, fungal or protozoan. Common symptoms of such infections include fever, confusion and increased heart and breathing rates. A low blood pressure caused by sepsis that does not improve after fluid replacement is called "septic shock". (Weis *et al.*, 2017).

Sepsis has been estimated to be responsible for over 50 % of the deaths in the ICUs in the United States (Paoli *et al.*, 2018). The mortality rate of sepsis varies by how severe the disease has gotten, but also by geographic location. The reported mortality rates are between 22% and 76% (Levy *et al.*, 2012). The disease is also deemed to be the highest cost burden to the United States hospital system, with estimates from over 24 billion (Paoli *et al.*, 2018) to more than 41.5 billion US dollars (U.S. Department of Health & Human Services, 2020). The number is also expected to increase with greater speed than what the aging population should be responsible for (U.S. Department of Health & Human Services, 2020).

The exact pathobiology behind sepsis is still unknown to date. This means, there is no gold standard diagnostic test available to determine if a patient has sepsis, and medical professionals rely on their judgement of a body's response to diagnose a patient with sepsis. The lack of pathobiological knowledge and a diagnostic test have led to the evolution of sepsis definition over time with three iterations. Sepsis was defined for the first time in 1991, and second time in 2001. These definitions are known as Sepsis-1 and Sepsis-2, respectively. The latest definition of Sepsis was published in 2016 by The Third International Consensus Definitions for Sepsis and Septic Shock. It is known as Sepsis-3. (Weis *et al.*, 2017)

Sepsis-3 definition

Sepsis-3 is the latest definition of sepsis published in 2016 and defines sepsis as life-threatening organ dysfunction due to dysregulated host response to infection. It uses the SOFA score as a clinical tool to diagnose a patient with a suspected infection of having sepsis. Sepsis-3 defines

sepsis in a patient with suspected infection, when the SOFA score increases by two or more points. (Weis *et al.*, 2017)

The time for the SOFA score increase to occur is not defined by Weis et al (2017), but one commonly used window is 24 hours for the SOFA score change so that it occurs between the period that is 48 hours before and 24 hours after the suspected infection time. This definition was used in a Sepsis-3 related study by Seymour *et al.* in 2016. Different target definitions are examined more closely in Chapter 4.1.

3 MACHINE LEARNING

Machine Learning was first defined by Arthur Samuel in 1959. His work on a machine playing checkers at IBM Poughkeepsie Laboratory created an algorithm that learned to play better by evaluating moves of several games it played against itself (Samuel, 1959). A machine can be said to be learning, when it can change its behaviour to be more optimal regarding the task at hand by evaluating its own previous decisions against a performance metric (Mitchell, 1997).

Tasks for machine learning algorithms can be of various type like regression and classification. In classification the task of the algorithm is to determine into which predefined class an unseen sample of data belongs. The algorithms can also be separated into two main classes: supervised and unsupervised learning. In supervised learning the true labels of samples are known, and the algorithm is taught to predict them from the input. In unsupervised learning, there are no known labels for the data and the algorithm is used to find yet unknown patterns or relations from the data. In this thesis, the focus is on classification tasks, as the problem is to classify patients to either be septic or not. The presence of labels or classes indicates that the problem is of supervised type. (Alpaydin, 2014)

This chapter describes the process behind a learning algorithm, in particular the phases included in this process and affecting its performance. Three classical ML algorithms, Logistic Regression, Random Forest and Gradient Boosting, are introduced. Also more complex algorithms, such as Artificial Neural Networks, and their subtypes are examined.

3.1 Training and Evaluation

The process of iterating a ML model through known samples and changing its parameters according to feedback it receives from true and false predictions is called training (Bishop, 2006). It is important to be able to evaluate the model's performance on out of sample data without bias. To make this possible, the dataset is often divided into three subsets:

- **Training set –** a set used for training a classifier
- **Validation set** – a set of data used to evaluate the classifier and tune its parameters

- **Testing set –** a set of data used only to evaluate the performance of the developed classifier

The training and validation set together form the development set. Using the development set, the model is trained and its hyperparameters are tuned. The model's classification performance is then evaluated against the testing set, that has been withheld from the model during the training process. This way, unbiased performance metrics can be calculated for the classifier. The division of data into the subsets is illustrated in Figure 1. (Bishop, 2006)

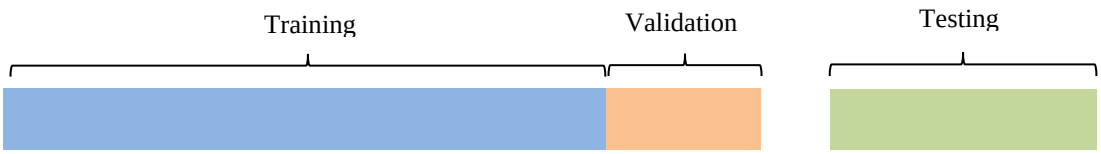

Figure 1 – Splitting data to training, validation and test set.

3.2 Cross-Validation

In applications with limited available data, a single split to training and validation to train the model will struggle to deliver good results. Cross-validation is used to get statistically more reliable results. One way of using cross-validation is splitting the development set into *K* subsets. The training is performed so that it is repeated *K* times where each time one subset is taken as a validation set and the rest of the data as the training set. This is done for every possible combination within the subsets. (Bishop, 2006)

Using cross-validation gives a better estimate of a model's predictive performance for out-of-sample data, as the uncertainty is decreased by using more data validating it and overfitting to the hyperparameters is avoided. Using multiple training runs with different validation sets means a longer computation time, that increases with the factor of *K* compared to the single split method. Figure 2 illustrates a 5-fold cross-validation split. (Bishop, 2006)

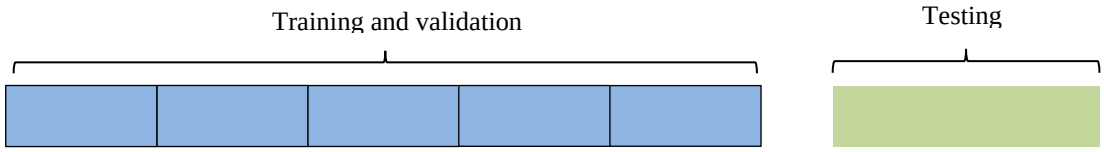

Figure 2 – 5-fold cross-validation split.

3.3 Performance Metrics

A binary classification has two possible outcomes within the predicted variable. This means that there are four different possible outcomes. A correctly predicted positive sample is a True Positive (TP), while an incorrectly predicted positive sample is a False Negative (FN). On the contrary, a correctly predicted negative sample is a True Negative (TN), and an incorrectly predicted negative sample a False Positive (FP). (Alpaydin, 2014)

Table 2 – A confusion matrix of a binary classification problem where actual positives and negatives are the correct classes and the predicted positives and negatives are the classes predicted by the model.

	Predicted positive	Predicted negative
Actual positive	True Positive (TP)	False Negative (FN)
Actual negative	False Positive (FP)	True Negative (TN)

Using the outcomes in Table 2, multiple performance metrics can be calculated. These performance metrics are used to evaluate the goodness of the model fit and its performance on the testing set. The different metrics for the classifier evaluation are listed in Table 3. The most widely used metric is Accuracy. Accuracy simply states how many samples the model was able to classify correctly. In a dataset with unbalanced distribution of outcomes a good accuracy score can be obtained just by always predicting the majority class. Many datasets in the medical domain suffer from a small set of positive cases, which is considered the more interesting class for prediction, so that the accuracy is not always a sufficient metric for evaluating model performance.

To capture the model's performance on the positive class, the True Positive Rate (TPR) metric can be used. It is also known as sensitivity and recall. TPR tells how big proportion of actual positive samples the model was able to correctly classify. The True Negative Rate (TNR), also known as specificity, is a measure that tells how many of the actual negative samples were correctly classified as negative. The False Positive Rate (FPR) and False Negative Rate (FNR) can be calculated in similar fashion.

The Positive Predictive Value (PPV), also known as precision, measures the ratio of true positives to all samples that the model predicted as positive. The Negative Predictive Value (NPV) measures the ratio of actual negatives within all samples that model predicted as negative. (Alpaydin, 2014)

Table 3 – A set of performance metrics for binary classification. These metrics require a fixed threshold dividing the predictions into two classes.

Performance metric	**Formula**
Accuracy	$\frac{\mathrm{TN}+\mathrm{TP}}{\mathrm{TN}+\mathrm{FP}+\mathrm{FN}+\mathrm{TP}}$
Prevalence	$\frac{\mathrm{TP}}{\mathrm{TN}+\mathrm{FP}+\mathrm{FN}+\mathrm{TP}}$
True Positive Rate (TPR), sensitivity, recall	$\frac{\mathrm{TP}}{\mathrm{FN}+\mathrm{TP}}$
True Negative Rate (TNR), specificity	$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$
False Positive Rate (FPR)	$\frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}$
False Negative Rate (FNR)	$\frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}}$
Positive Predictive Value (PPV), precision	$\frac{\mathrm{TP}}{\mathrm{FP}+\mathrm{TP}}$
Negative Predictive Value (NPV)	$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}}$
Positive Likelihood Ratio (LR+)	$\frac{\mathrm{TPR}}{\mathrm{FPR}}$
Negative Likelihood Ratio (LR-)	$\frac{\mathrm{FNR}}{\mathrm{TNR}}$
F1-score	$2 \times \frac{\mathrm{PPV} \times \mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}}$

where TP = True Positive, FN = False Negative, FP = False Positive, TN =True Negative

Using the previously described metrics, additional ones can be calculated. The Positive Likelihood Ratio (LR+) tells the ratio of how much more likely it is to classify an actual positive correctly as positive, than an actual negative falsely as positive. The Negative Likelihood Ratio (LR-) on the other hand tells how much more likely it is for an actual positive to be falsely classified as negative, than for an actual negative to be correctly classified as negative. These values are widely used in diagnostics since the classifier can be judged to perform better the higher LR+ is, and the lower the LR- value is. A combination score, the F1-score, is a weighted average of precision and recall. (Alpaydin, 2014)

Some ML algorithms' initial output is a numeric value, often representing a probability. In binary classification, such an algorithm decides the class that the sample belongs to based on the probability it calculates. To perform the decision, the algorithm compares the value to a threshold value and assigns samples that obtain a value lower than the threshold to one class and values that obtain a value higher than the threshold to another. As the value of the threshold influences how the model predicts, changing the threshold value also changes the performance metrics listed in Table 3. As setting the threshold value is not always a trivial task, the performance metrics' dependence on it makes a comparison between two different ML models using only common metrics difficult. A way to compare the threshold's effect on the model performance is to use a threshold-free measurement. The two most common ones are the Receiver Operating Characteristics (ROC) curve and the Precision Recall Curve (PRC). (Alpaydin, 2014)

Both ROC and PRC are used by plotting and visually inspecting the curve to determine an optimal threshold for classification. The ROC curve is a plot of TPR (sensitivity, recall) against FPR (1 – specificity) that captures the sensitivity and specificity of the classifier with all possible cut-offs for a threshold value. The Precision-Recall curve is a plot of precision (PPV) against recall (TPR, sensitivity), as the name suggests. A classifier's performance can be calculated by using the Area Under the Receiver Operating Characteristics (AUROC) metric. A totally random classifier has a ROC curve going straight from the bottom left (0,0) corner to the upper right (1,1) corner on a plot of ROC. The value of AUROC is calculated as an area of the space that is left underneath the ROC curve on the plot. This means it will have an AUROC value of 0.5, which can be held as a baseline representing coin toss performance for classifying

samples to two classes. A classifier that manages to correctly classify all the samples has an AUROC value of 1. (Alpaydin, 2014)

For data sets with unbalanced distribution between classes, the AUROC curves can fail to deliver information on the difference of the classifier's performance on the classes. Often one of the classes is of more interest, but sensitivity and specificity scores are affected the same no matter where the misclassifications happen. An example of such problem is often present in medical domain, where a patient with disease diagnosed as healthy is more problematic than classifying a healthy patient as sick. An alternative to AUROC curve is the Area Under Precision Recall Curve (AUPRC), which is the area of a plot left under PRC curve. A totally random classifier would have AUPRC performance equal to the prevalence of positive class in the dataset. Dataset where 5% of the samples are positive, the AUPRC value is 0.05 for random classifier. If the dataset would have 40% prevalence of positive samples, AUPRC is then 0.4 for random classifier. (Saito and Rehmsmeier, 2015)

The informatic value of AUROC and AUPRC is shown to differ depending on the uniformity of class distribution between samples. A classifier built for data with even distribution of outcome classes is easier evaluated by ROC curve and AUROC value. For classifier built using data with uneven class distribution, the threshold and performance is easier evaluated using PRC curve and AUPRC value. In this thesis, both AUROC and AUPRC are used for performance evaluation of the models.

3.4 Logistic Regression

Logistic Regression (LR) is a binary classification method. LR gives as an output a probability of a given input of belonging to a class. It assumes that the classes are linearly separable, and gives the probability output using the formula:

$$p(c = 1 \,|\, \mathbf{x}) = \frac{1}{1 + \exp\left[-(\mathbf{w}^T\mathbf{x} + b)\right]} = \hat{y} \qquad (1)$$

$$p(c = 0|\mathbf{x}) = 1 - p(c = 1|\mathbf{x}) = 1 - \hat{y}, \tag{2}$$

where **x** is the input vector, **w** is the weight vector and b the bias term. The term $\hat{y}$ is the model's output, a probability of sample belonging to class 1. In the functions above, the positive class is $c = 1$ and negative class $c = 0$. (Alpaydin, 2014) The model maps the projection of linear equation $z = \mathbf{w}^T\mathbf{x} + b$ using a sigmoid function:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}\,, \tag{3}$$

that limits the output into space [0, 1]. The most common way to train the LR classifier is using maximum likelihood to solve for the parameters **w** and b. The likelihood of samples $\mathbf{x} = [x_0, x_1, \dots, x_{n-1}]$ to get class labels $\mathbf{y} = [y_0, y_1, \dots, y_{n-1}]$ can be written as:

$$L(\mathbf{w}, b|\mathbf{x}) = \prod_{n=1}^{N} p(c = \mathbf{y}_n \,|\, \mathbf{x}_n; \mathbf{w}_n, b)\,. \tag{4}$$

To simplify the equation, a negative log-likelihood can be taken from the above equation:

$$-\ln\big(L(\mathbf{w}, b|\mathbf{x})\big) = -\sum_{n=1}^{N} \ln\left(p(c = \mathbf{y}_n \,|\, \mathbf{x}_n; \mathbf{w}_n, b)\right). \tag{5}$$

The log-likelihood equation can be rewritten using the class labels y and prediction probabilities $\hat{y}$ for each input:

$$\ln\big(L(\mathbf{w}, b|\mathbf{x})\big) = -\sum_{n=1}^{N} (y_n \ln(\hat{y}_n) + (1 - y_n) \ln(1 - \hat{y}_n)). \tag{6}$$

By minimizing the negative log-likelihood from equation 7 the likelihood in equation 6 gets maximized. The equation of negative log-likelihood can be used to solve parameters **w** and b:

$$\mathbf{w}^*, b^* = \underset{\mathbf{w},\mathrm{b}}{\operatorname{argmin}} \left(-\sum_{n=1}^{N} (y_n \ln(\hat{y}_n) + (1 - y_n) \ln(1 - \hat{y}_n)) \right). \quad (7)$$

The loss function between the label y_n and model output $\hat{y}_n$ can be written as:

$$L(\hat{y}_n, y_n) = -(y_n \ln(\hat{y}_n) + (1 - y_n) \ln(1 - \hat{y}_n)). \quad (8)$$

The above equation can be split depending on the value of the label y_n:

$$L(\hat{y}_n, y_n) = \begin{cases} -\ln(\hat{y}_n), & y_n = 1 \\ -\ln(1 - \hat{y}_n), & y_n = 0. \end{cases} \quad (9)$$

From above equation an approximation error can be calculated by taking the average of each input's loss:

$$E(\mathbf{w}, b) = \frac{1}{N} \sum_{n=1}^{N} L(\hat{y}_n, y_n) = -\frac{1}{N} \sum_{n=1}^{N} (y_n \ln(\hat{y}_n) + (1 - y_n) \ln(1 - \hat{y}_n)). \quad (10)$$

The optimization problem shown in equation 10 has no closed form solution. An iterative gradient descent approach can be used to solve the minimization problem. First, the parameters of the model are initiated randomly. After each iteration, they are updated with small steps to the opposite direction of the gradient of the error function with respect to the model parameters:

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \nabla_{\mathbf{w}} E(\mathbf{w}, b) \quad (11)$$

$$b \leftarrow b - \lambda \nabla_b E(\mathbf{w}, b), \quad (12)$$

where λ is the learning rate defining step size and ∇ operator denotes the gradient. (Alpaydin, 2014)

3.5 Decision Trees

A decision tree is a method of nonparametric estimation, as compared to the LR model that was a type of parametric estimation. In these nonparametric models the input space is divided into non-overlapping local regions defined by some distance measure like Euclidean norm. As the task in this thesis is binary classification, the focus is only on classification trees. It is also assumed that the input variables are of numeric form. The decision tree divides the feature space into regions and assigns to each region a probability of belonging to class C_1, or a direct label $\{0, 1\}$. (Alpaydin, 2014)

A single decision tree can be expressed in general form:

$$\hat{y} = P_{tree}(C_1|x), \tag{13}$$

representing the probability of $\boldsymbol{x}$ belonging to class C_1.

The regions into which the data is divided are determined using binary splits of the input data. The target of each split is to divide the input data as purely as possible, using some measure of purity. An optimal situation, meaning a pure split, occurs when the input data can be split into two regions which each only contain instances belonging to a single class, C_0 or C_1. A decision node defining two subregions consists of a single split. The decision tree rarely accomplishes a pure split on the input data after a single split. Each of the two subregions defined by the first split can be split further in a similar fashion, adding the number of subregions to four. This process is done recursively until the tree accomplishes a pure split, or a stopping criterion is met. The nodes of decisions form a tree, hence the name decision tree. The first node is called the root node, and all the nodes with no successors are called leaf nodes, or leaves in short. All nodes that are not root or leaf nodes are referred to as internal nodes. (Alpaydin, 2014)

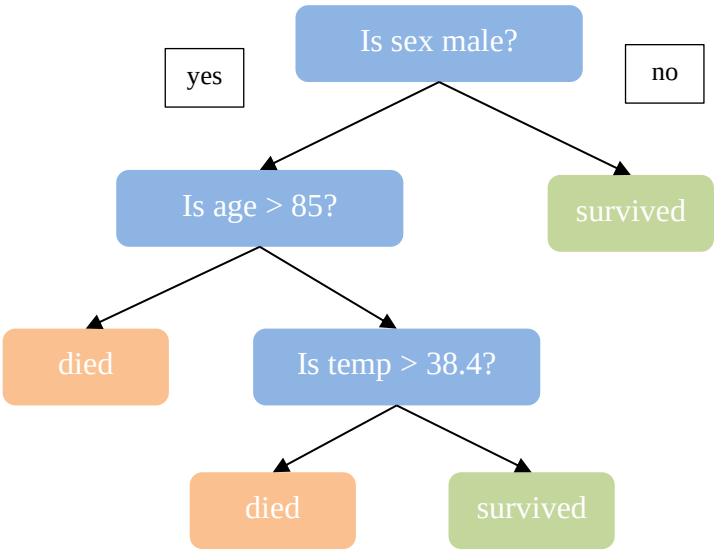

Figure 3 – An example of a decision tree illustrated.

In the split described above, a decision node j of the tree performs a test on one of the features in the input data, $d = 1, \dots, D_{input}$. A threshold value τ_j is determined based on the feature's values, based on which the split is done in the node. All data instances with a value below the threshold value τ_j are assigned to the first subregion, and with values above to the second one. Given region $\mathbb{X}_j \subseteq \mathbb{X}$ at node j, the split in a node to two subregions is formally performed as:

$$\begin{cases} \mathbb{X}_{L_j} = \{x \mid x_d < \tau_j\} \\ \mathbb{X}_{R_j} = \{x \mid x_d \geq \tau_j\}, \end{cases} \quad \mathbb{X}_j = \mathbb{X}_{L_j} \cup \mathbb{X}_{R_j}, \mathbb{X}_{L_j} \cup \mathbb{X}_{R_j} = \emptyset, \tag{14}$$

where $\mathbb{X}_{L_j}$, $\mathbb{X}_{R_j}$ are the subregions resulting from node j split.

Each region can be given either a probability or a label. This applies to all the nodes in the tree, not just leaves. The probability of a node belonging to the class C_1 can be estimated as the ratio of positive instances that arrive to the node:

$$\hat{p}_j(C|x) = \frac{N_{j,1}}{N_j}, \tag{15}$$

where $N_{j,1}$ is the number of positive instances coming to node j, and N_j the total number of data instances coming to the node. Growing the trees until all leaves achieve a pure split will give the nodes a direct label, probability being $\hat{p} = \{0, 1\}$. Growing trees using another stopping criterion than achieving pure splits will result into the leaf nodes having a probability output. Whether the decision tree outputs labels or probabilities is a question of design.

Choosing the variables that occur in the upper nodes of the tree to split the data first is important for keeping the tree size small. A small tree is computationally less demanding in the prediction phase, generalizes better to unseen data and provides better informative value about the task, assuming it is not too small to underfit the data (Alpaydin, 2014). Achieving an optimal split needs a measure of impurity, where a common choice is information gain using entropy (Alpaydin, 2014). An event with low probability should have high entropy, and high probability a low entropy. Entropy can be defined as:

$$H(T) = I_E(p_1, p_2, \dots, p_n) = -\sum_{j=1}^{n} p_j \log_2 p_j \,, \tag{16}$$

where $p_1, p_2, \dots, p_n$ are fractions adding up to one, that represent the percentage of each class being present in a child node that results from a split in the tree (Alpaydin, 2014). Gini index, another common selection criterion for splits, is the probability of a randomly sampled point to be misclassified in a node. The formula of Gini index of a leaf node is:

$$I_{gini}(T, x) = \sum_{i=1}^{2} p_i(1 - p_i) \,, \qquad p_1 + p_2 = 1. \tag{17}$$

The impurity of a decision node is the average of its sub nodes weighted by the number of data points belonging to each branch:

$$I_{node} = -\frac{N_{\text{left}}}{N} I_{\text{left}} - \frac{N_{\text{right}}}{N} I_{\text{right}}, \tag{18}$$

where N is the total number of data points, N_{left} and N_{right} are the numbers of datapoints in the left and right sub nodes, respectively. I_{left} and I_{right} are the impurities of the corresponding sub nodes. By maximizing the Gini index of a node, the optimal splits can be defined. (Alpaydin, 2014)

3.6 Random Forest

Random Forest is an ensemble learning algorithm based on Decision Trees. As the name suggests, instead of generating a single decision tree, a multitude of trees are trained. For each tree, a bootstrap sample from the data is taken instead of using the whole dataset. Also, only a subset of all available input variables is used to generate the single trees of the forest. The training of a single tree is performed in a similar manner as described in Chapter 3.5.

The forest makes predictions by performing a vote among all of its trees. In the binary classification example, each tree is fed with the input data and then gives out the class it assumes to be the most likely. The tree then casts this outcome as its vote. The final output of the forest is the outcome with most votes after aggregating through all trees' votes (majority vote). A probability of the classification can be calculated by dividing the vote count by total number of trees in the forest. The random forest tends to perform better by not overfitting as easily on a dataset as a single decision tree, as the randomness is an inherent part of how the method works. (Breiman, 2001)

3.7 Gradient Boosting

Gradient boosting (GB) is another type of ensemble machine learning algorithm. It uses a collection of base learners to assemble a stronger model. The fitting of the base learners happens one at a time in the GB algorithm, and these incremental improvements are called boosts. The sum of the fitted base learners becomes the final predictor. (Chollet, 2017)

In each boosting round n, a base learner $h(x, a)$ with parameters a gets fitted. The learner multiplied by β_n is added to the previous model $\hat{f}_{n-1}$. The fitting at the nth round corresponds to solving the minimization problem equation

$$(\beta_n, a_n) = \arg\min_{\beta,a} \sum_{i=1}^{N} L(y_i, \hat{f}_{n-1}(x_i) + \beta h(x_i, a)), \tag{19}$$

where L is the employed loss function. The new estimator becomes

$$\hat{f}_n(x) = \hat{f}_{n-1}(x) + \beta_n h(x, a_n). \tag{20}$$

The gradient boosting algorithm assumes the direction of negative gradient to be the best estimate for the new estimator. A learner most parallel to the negative gradient is chosen, as the base learner $h(x, a)$ itself doesn't necessarily match the gradient. The approximation is then updated by estimated optimal step-size into this direction.

Gradient tree boosting, one of many gradient boosting methods, is a method using decision trees as weak learners. During each boosting round, a decision tree is constructed. The tree can be represented in form

$$f(x) = \sum_{i=1}^{N_{leaf}} w_i \mathbb{I}_R \quad (x), \tag{21}$$

where w_i is the output of the tree in leaf i, R_i are the disjoint regions covering the leaves of the decision tree and N_{leaf} is the number of leaves in the tree. The optimal weights of the tree are calculated using a selected loss function. (Chen and Guestrin, 2016)

To avoid overfitting, regularization is often applied to equation 20. Most common options are to use l_1 and l_2 regularizations, that add term $\|w\|_1$ or $\|w\|_2^2$ to equation (20) correspondingly (Alpaydin, 2014). The strength of the regularization is controlled by multiplying each regularization term with a constant. Another way to do regularization is to apply shrinkage to the term β_n by multiplying it with a constant ν between 0 and 1. The shrinkage constant ν is also known as learning rate. Equation 20 then becomes

$$\hat{f}_n(x) = \hat{f}_{n-1}(x) + \nu\beta_n h(x, a_n). \tag{22}$$

3.8 Artificial Neural Networks

Deep learning and Artificial Neural Networks (ANNs) are a part of machine learning that has its inspiration from biological neural networks. They have gained popularity in the recent years due to their improved capabilities of handling complex relationships between variables. Widely applied areas are, e.g. computer vision and speech recognition (Alpaydin, 2014). This chapter introduces basic concepts behind deep learning and introduces methods behind convolutional and recurrent neural networks that are used in this study.

3.8.1 Neural Network Basics

The Neural Network models consist of many mathematical operation units called neurons. Each neuron receives an input and maps it to an output by multiplying the input with weights and adding a bias. The next step is to perform affine transformation, where the neuron's output is turned nonlinear using an activation function. The equation for affine transformation is:

$$o(x_n) = \phi(\mathbf{w^T}_n \mathbf{x}_n + b), \tag{23}$$

where ϕ is an activation function, x_n is the feature vector, $\mathbf{w}_n$ are the weights and b the bias term. The weights give a relative importance to each input arriving to the neuron. The bias further adjusts the output of the neuron by increasing or decreasing the output value before passing it to the activation function. The output resulting from a combination of the neuron and the activation function is then passed forward to the next neurons, and thus works as an input for it. (Goodfellow, Bengio and Courville, 2016)

Multiple neurons in the ANN form structures called layers, as represented in Figure 4. The equation 23 can be expanded to the whole layer using matrix representation:

$$o(x) = \phi(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{24}$$

where $\mathbf{x}$ is the layer's input with size n, $\mathbf{W}$ is the weight matrix of size $m \times n$, n being the input size and m the number of neurons in the layer. The matrix multiplication on $\mathbf{x}$ and $\mathbf{W}$ results

into a vector of size m. The vector **b** represents the bias for each neuron in the layer and is of size m. Finally, an activation function ϕ is applied elementwise, resulting in the layer's output vector of size m. (Goodfellow, Bengio and Courville, 2016)

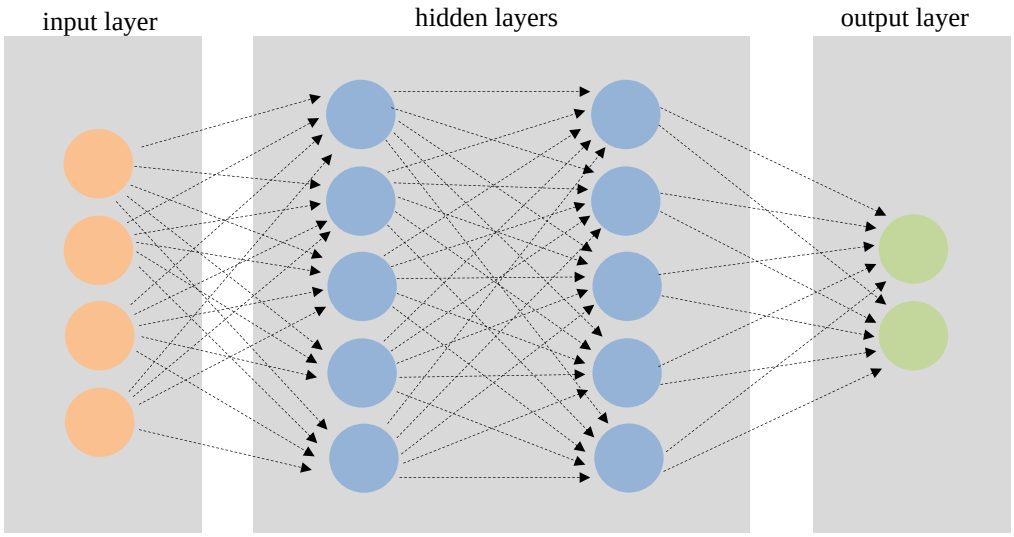

Figure 4 – An example of a NN with an input layer, two hidden layers and an output layer. The input layer has four features, the hidden layers five neurons each and the output layer two possible outcomes. (Stanford Vision and Learning Lab, 2019)

3.8.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a type of neural networks developed especially for sequential data. Some examples of sequential data are text and audio, that can be used for problems like speech recognition. The better performance of RNNs over basic neural networks comes from their ability to process sequences of various lengths. (Alpaydin, 2014)

A RNN cell takes a timestep of input data together with the feedback of previous outputs at a time. The feedback of the previous outputs is called the hidden state. A structure of RNN is represented in Figure 5.

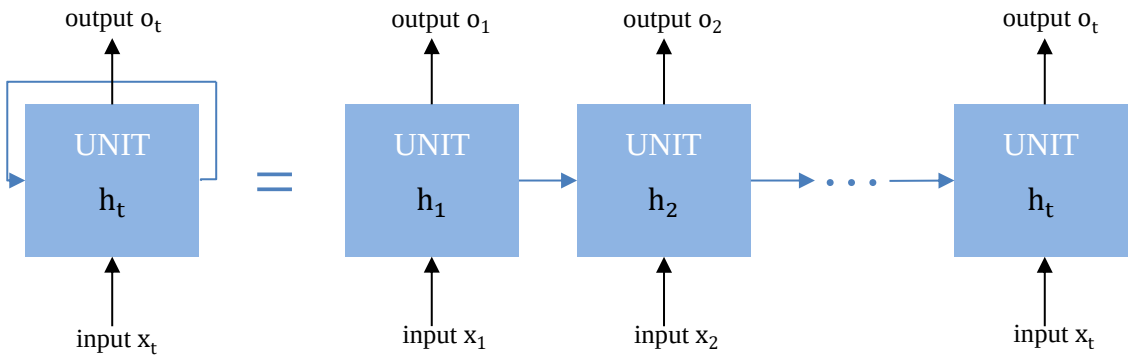

Figure 5 – An illustration of RNN structure. (Olah, 2015)

In each timestep, two equations occur in an RNN unit:

$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}) \tag{25}$$

$$\mathbf{o}_t = \phi(\mathbf{V}\mathbf{h}_t + \mathbf{c})\,, \tag{26}$$

where $\mathbf{x}_t$ is the input at the timestep t and $\mathbf{W}$ is the weight matrix associated to it. These are combined with the previous hidden state $\mathbf{h}_{t-1}$ and the weight matrix $\mathbf{U}$ associated**.** Term $\mathbf{b}$ represents the bias added. The term σ represents the activation function, most commonly a tanh activation function, which is used to calculate the new hidden state $\mathbf{h}_t$. The unit's output $\mathbf{o}_t$ is calculated as a regular affine transformation using the hidden state $\mathbf{h}_t$, weight matrix $\mathbf{V}$, bias $\mathbf{c}$ and an activation function, represented as ϕ in the equation. (Alpaydin, 2014)

The RNNs can have different structures with a varying combination of inputs and outputs. The RNN represented in Figure 6 produces an output at each of the timesteps and is called many-to-many. Another recurrent structure, called many-to-one, reads the whole sequence and produces one output. Any combination of inputs and outputs mapped to each other is possible making recurrent networks adaptable for many sequence problems. In Figure 6 there are examples of different mappings of inputs to outputs in RNNs.

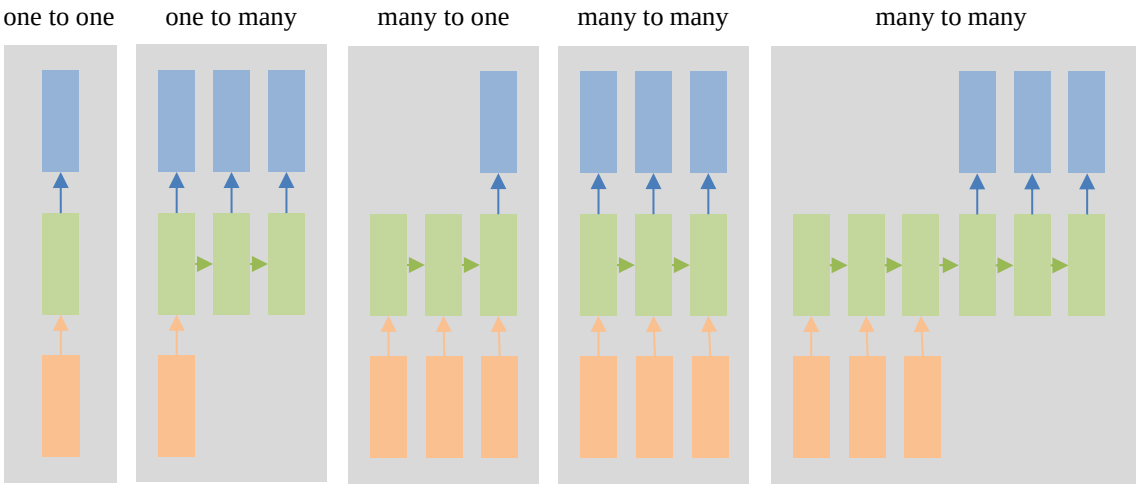

Figure 6 – Illustrated examples of different RNN structure combinations mapping inputs to outputs. From left to right: one-to-one, one-to-many, many-to-one and two different many-to-many mappings. (Karpathy, 2015)

The RNNs are often used on problems with long input sequences that result to many timesteps during the training. This results in vanishing magnitude of the gradient when going further back in backpropagation and causes the RNNs to fail in long term dependencies. Special RNNs have been designed, called gated RNNs, to tackle the vanishing gradient problem. The following section describes one called Long Short-Term Memory (LSTM). (Goodfellow, Bengio and Courville, 2016)

Long Short-Term Memory

The Long Short-Term Memory (LSTM) recurrent neural network structure was developed by Hochreiter and Schmidhuber (1997). The LSTM structure includes some extensions to the classic RNN structure and introduces self-loops to allow the gradient to flow for long distances, tackling the vanishing gradient problem. A LSTM unit operates and can be used the same way as any RNN unit, but the calculations happening inside it differ a bit. In addition to the input $\mathbf{x}_t$ and the hidden state from previous timestep $\mathbf{h}_{t-1}$, the LSTM unit also takes in the previous timestep's internal state $\mathbf{c}_{t-1}$. Correspondingly, the LSTM cell also outputs two states: a new hidden state $\mathbf{h}_t$ and a new internal state $\mathbf{c}_t$. (Hochreiter and Schmidhuber, 1997)

The LSTM unit includes three gates called the input gate $\mathbf{i}_t$, the forget gate $\mathbf{f}_t$ and the output gate $\mathbf{o}_t$. Each gate has its own weight matrix and the calculation is based on a similar equation as in the basic RNN unit:

$$\mathbf{i}_t = \sigma(\mathbf{U}^i\mathbf{x}_t + \mathbf{W}^i\mathbf{h}_{t-1} + \mathbf{b}^i) \quad (27)$$

$$\mathbf{f}_t = \sigma(\mathbf{U}^f\mathbf{x}_t + \mathbf{W}^f\mathbf{h}_{t-1} + \mathbf{b}^f) \quad (28)$$

$$\mathbf{o}_t = \sigma(\mathbf{U}^o\mathbf{x}_t + \mathbf{W}^o\mathbf{h}_{t-1} + \mathbf{b}^o)\,. \quad (29)$$

The equations use a sigmoid activation function that scales the vectors to the range between 0 and 1, and work as a gate in the unit. Multiplying these gates elementwise with other vectors defines how much information from the vector is fed further. (Hochreiter and Schmidhuber, 1997; Goodfellow, Bengio and Courville, 2016)

The LSTM unit's three states are the input state $\mathbf{g}_t$, the internal state $\mathbf{c}_t$ and the hidden state $\mathbf{h}_t$. The states are updated using the following equations:

$$\mathbf{g}_t = \tanh(\mathbf{U}^g\mathbf{x}_t + \mathbf{W}^g\mathbf{h}_{t-1} + \mathbf{b}^g) \quad (30)$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \circ \mathbf{f}_t + \mathbf{g}_{t-1} \circ \mathbf{i}_t \quad (31)$$

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \circ \mathbf{o}_t\,, \quad (32)$$

where $\circ$ is elementwise multiplication. The input gate $\mathbf{i}_t$ defines how much of the computed new input state $\mathbf{g}_t$ is let through. Correspondingly, the forget gate $\mathbf{f}_t$ defines how much of the previous internal state $\mathbf{c}_{t-1}$ is let through. Finally, the output gate $\mathbf{o}_t$ defines how much of the new internal state $\mathbf{c}_t$ is let through forming the hidden state $\mathbf{h}_t$. This hidden state $\mathbf{h}_t$ is then exposed to higher layers in the network and is passed together with the internal state $\mathbf{c}_t$ to the next timestep. The gating explained above is presented in Figure 7. (Hochreiter and Schmidhuber, 1997; Goodfellow, Bengio and Courville, 2016)

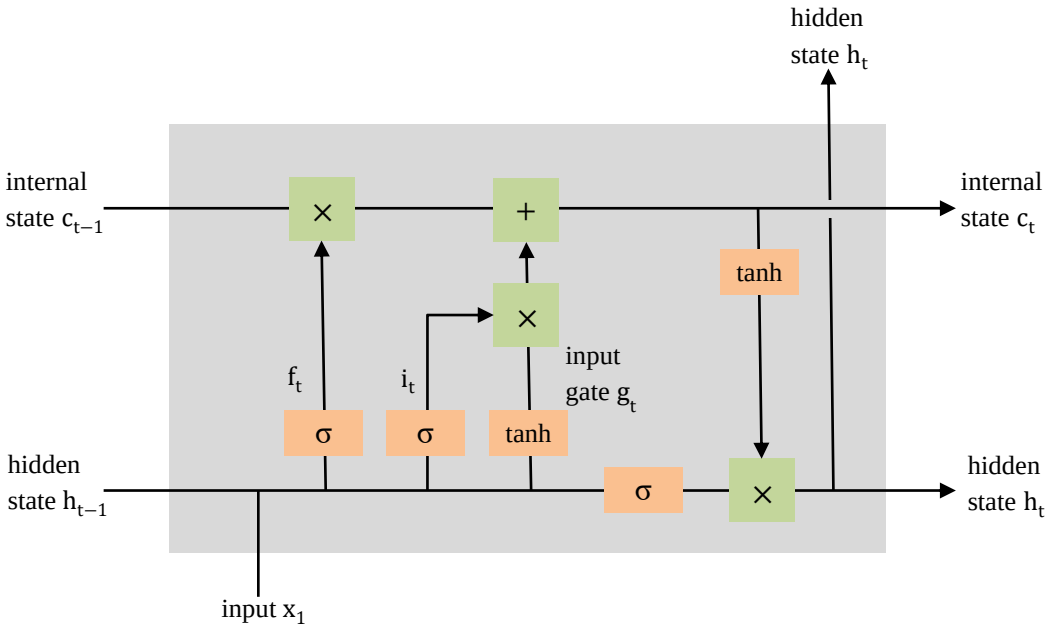

Figure 7 – An illustration of the LSTM unit and its gating. (Olah, 2015)

4 RELATED WORK

The prediction of sepsis is one of the most studied subjects in the field of machine learning for health care. This chapter introduces and examines related studies on the topic that have influenced this thesis work. As the definition of sepsis has changed over time the focus is in those related works that use the same Sepsis-3 definition as in this work.

4.1 Prediction Targets

The review study done by Fleuren *et al.* (2020) demonstrates differences in target definitions between studies. Most studies using Sepsis-3 definition require suspected infection to appear as a combination of drawn blood culture and ordered antibiotics within a specified time period. If the culture is taken first, the antibiotics must be ordered within 72 hours. If antibiotics are first, the culture must be taken within 24 hours. The earlier event of the two marks the time of suspected infection, when the prerequisites are fulfilled. This follows the approach used by Seymour *et al.* (2016) in their work.

The time period in which the two-point change of SOFA score must occur varies between the studies. The most common time period for the SOFA score change to appear in is 24 hours. Other periods used are 48 hours and 72 hours, where the former usually comes from the requirement of the change occurring around the suspected infection. However, the 24-hour limit seems to be clearly more used than the other limits, and it was also used in the sepsis related challenge arranged by PhysioNet (Reyna *et al.*, 2020).

Another difference between studies is the time window around the suspected infection time for the SOFA score change to occur. The most commonly used limits are 48 hours before and 24 hours after, or 24 hours before and 12 hours after the suspected infection’s onset. While the first combination is used more often in the reviewed related works, there are some using the shorter window. An illustration of SOFA score changing with the first mentioned limits can be seen in Figure 8.

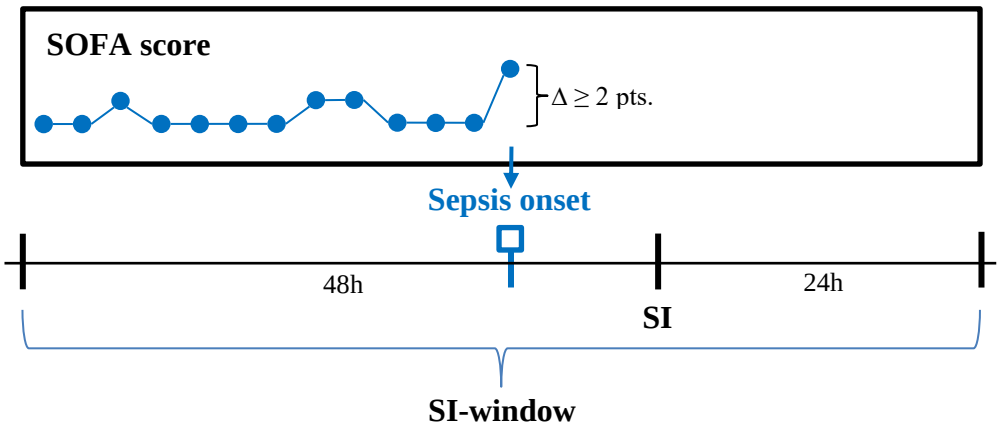

Figure 8 – Extracting sepsis onset using a 72-hour window around suspected infection time and a two-point change in the SOFA score. (SI = Suspected Infection) (Seymour *et al.*, 2016; Moor *et al.*, 2019)

The PhysioNet challenge takes an interesting approach in using time-dependent clinical utility functions for penalising correct positive predictions that happen too early or too late. The stated optimal prediction time used in the competition was 6 hours before the computed onset hour. The utility function rewards the classifications that happen close to the optimal prediction time the most, and the argument is that a correct positive prediction at this time is most valuable as the medical team can act on it early enough. Predictions that are more than 12 hours before or 3 hours after the onset are not clinically valuable as the actionability on the information decreases for the medical care team. The novel clinical utility metric demonstrated the drop in performance of some algorithms when used for predicting out-of-sample data better than AUROC and AUPRC according to the article on competition results. (Reyna *et al.*, 2020)

4.2 Input Variables and Models

The related studies have included many different approaches from using only a few commonly and frequently sampled vital signs (Mao *et al.*, 2018; Barton *et al.*, 2019) to using multiple vital signs, laboratory tests and patient demographics as input variables (Futoma *et al.*, 2017; Moor *et al.*, 2019). While including more variables usually tends to boost the model's performance, using a very large number of input variables can decrease the clinical interpretation and cause varying performance between hospitals that have different sampling practises for universally not so common measurements (Barton *et al.*, 2019).

In the review study by Fleuren *et al.* (2020), the use of input variables in different sepsis related studies was examined and can be seen in Figure 9. The most commonly used input variables are heart rate, respiratory rate, oxygen saturation and systolic and diastolic blood pressures. All these are routinely measured in the ICU and are called vital signs. Two patient demographics, age and gender, are also used in many papers on using ML for sepsis prediction. The most common laboratory tests used as input variables are white blood cell count, blood urea nitrogen, creatinine, platelet count, glucose and bilirubin. Apart from these mentioned, many others have been used, only not so commonly.

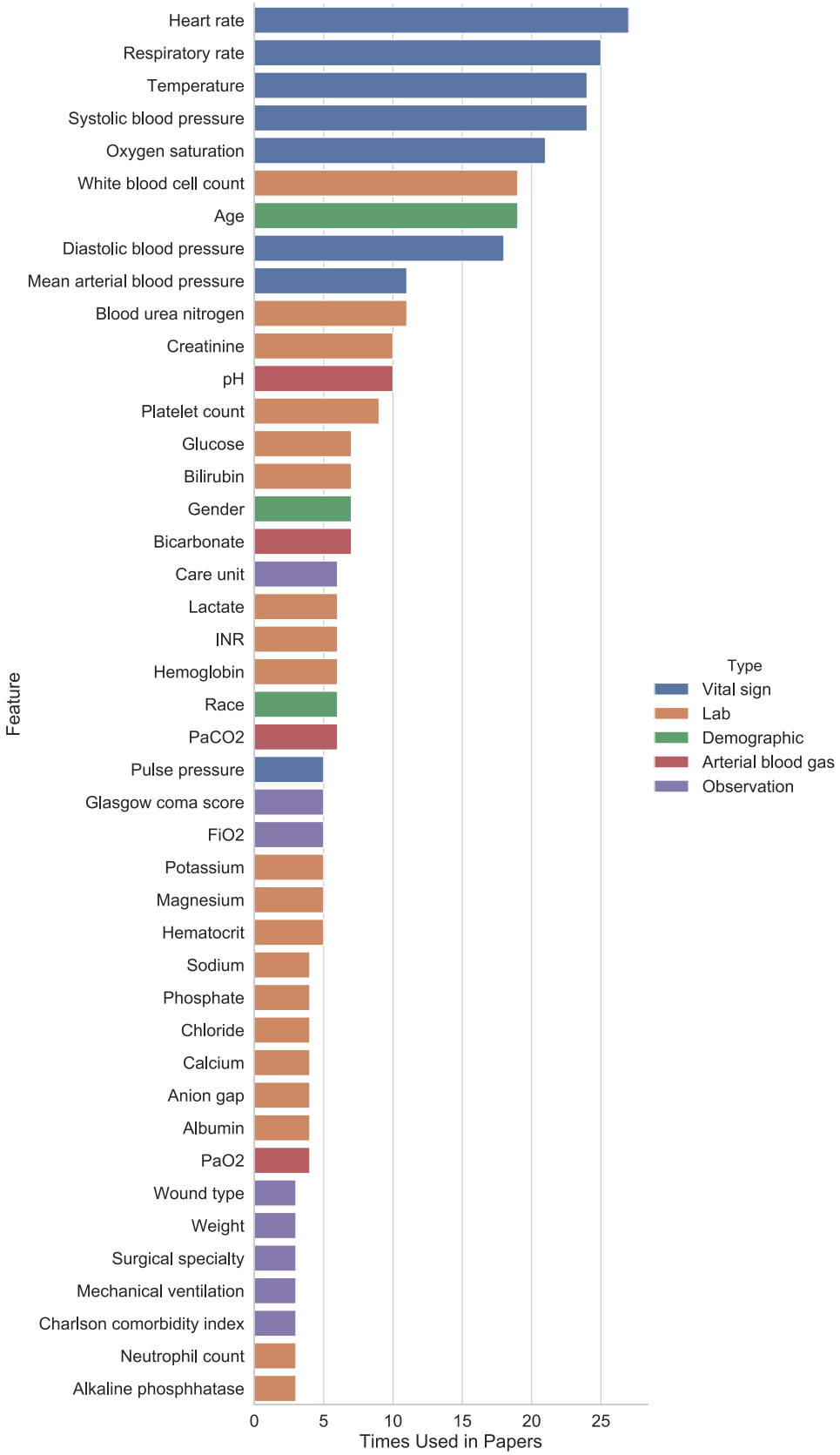

Figure 9 – Count plot representing the number of times a feature was used in the reviewed sepsis related ML studies in work by Fleuren *et al.* (2020).

The algorithms tested for predicting sepsis in the ICU include almost all most common classical and deep learning models available. This gives a good indication on what type of models seem to work the best for the problem at hand as demonstrated by related studies. One of the algorithms for predicting sepsis on the MIMIC dataset, called *InSight*, that gained more attention in the medical community, achieved AUROC of 0.92 at onset using a gradient tree boosting algorithm on six vital signs (Calvert *et al.*, 2016; Mao *et al.*, 2018). However, the algorithm used an older criterion of sepsis, having Systematic Inflammatory Response Syndrome (SIRS) criteria for its prediction target instead of sepsis-3 and is, thus, hard to compare against other methods.

The best performance of a sepsis prediction algorithm using sepsis-3 on MIMIC data that was found researching the related studies was done by Moor *et al.* (2019) using a combination of temporal convolution network for predicting and gaussian process adapter for imputing missing data. The best methods in the paper achieved the AUROC score of 0.9 at onset, but also mentions the achieved AUPRC score of 0.6 at the time of onset or 0.4 six hours before onset.

The PhysioNet's challenge on predicting sepsis is especially interesting for comparing the different models' goodness for predicting sepsis-3 labelled prediction targets in the ICU, even though the data is not the MIMIC. The three best ranked submissions include a signature based deep learning model (Morrill *et al.*, 2019), a gradient boosting model (Du, Sadr and De Chazal, 2019) and an XGBoost gradient boosting model (Zabihi, Kiranyaz and Gabbouj, 2019). Other models in top ten include mostly gradient boosting variants and some recurrent neural networks (Reyna *et al.*, 2020). The XGBoost method has also shown promising results on the MIMIC dataset by Barton *et al.*, (2019), only using limited amount of vital signs as input parameters.

5 DATA

This section describes the data used for the research in this thesis. Additionally, the patient demographics and input variables are examined. Finally, the inclusion criteria and study cohort used are presented.

5.1 Data Description

The MIMIC-III v1.4 dataset containing information on the patients' stays is collected at Beth Israel Deaconess Medical Center between years 2001 and 2012. The dataset contains vital signs, medications, laboratory measurements, observations and notes charted by care providers, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival data and more. The data is gathered from hospital admissions of 46 520 patients, who accounted for 61 532 visits to the ICU. (Johnson *et al.*, 2016)

The data has been collected using two different critical care information systems that were in place over the data collection period. The first system, Philips CareVue Information System was used during the years 2001 to 2008. The second system, iMDsoft MetaVision ICU took place and was used to gather the rest of the dataset. (Johnson *et al.*, 2016)

5.2 Patient Demographics

The mean age of the patients in the original dataset is 65.8 years and the median length of stay at the ICU is 2.1 days. On average there are 4579 charted observations and 380 laboratory measurements for each admission at the hospital. The distribution of patient demographics and hospital admission types can be seen in Table 4 and Table 5, respectively.

Table 4 – Patient demographics in the MIMIC dataset.

Demographic	**Number of patients (%)**
Total	46 520
Male	26 121 (56.2)
Female	20 399 (43.8)

Table 5 – Admission types in MIMIC dataset excluding neonatal patients.

Admission type	
Emergency	42071
Elective	7706
Urgent	1336

To prevent identifying patients, the ages of the patients with age over 89 years have been anonymised and appear as values of over 300 years in the dataset. These values are replaced with a value of 91.4 years for the algorithm as it is the median value within all the anonymised ages (Johnson *et al.*, 2016). In Table 6, the distribution of patients' age and length of stay (LOS) can be examined, after the age replacement has been done for oldest patients. The median age is 60.7 years in the processed dataset.

Table 6 – Distribution statistics of age and length of stay in MIMIC dataset.

Statistic	**Age (years)**	**Length of stay (days)**
Mean	53.1	4.9
Median	60.7	2.1
Standard deviation	27.0	9.6

Further, Figure 10 shows a histogram of the patient age distribution. It shows, that most patients are of age between 55 and 85 years. The number of patients with age less than 40 is quite small compared to the rest. The histogram of ICU stay LOS can be seen in Figure 11. The share of patients is clearly the highest in the beginning of the distribution, LOS between one and two days being the most common one. The longer the LOS the share of patients steeply decreases.

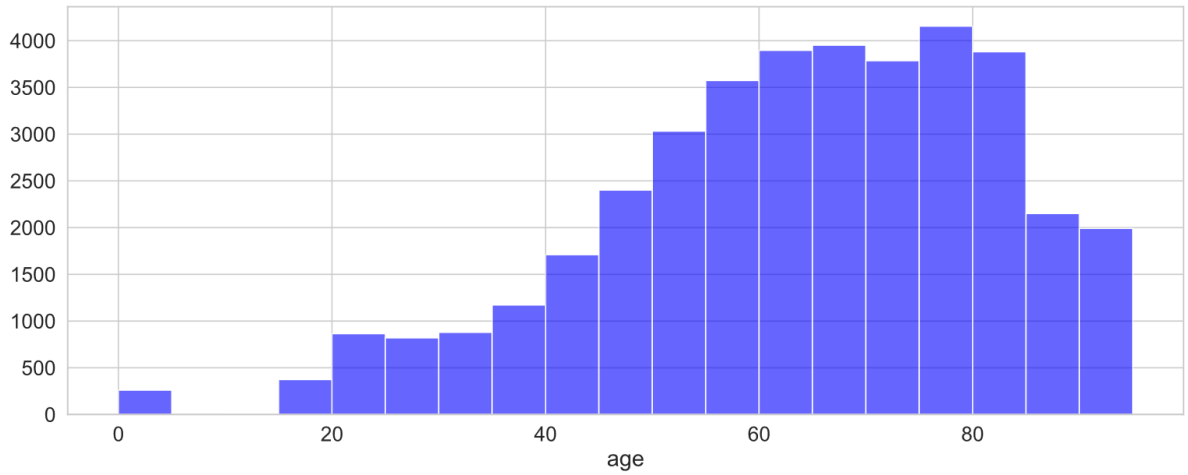

Figure 10 – Histogram of the age distribution of patients. Age for patients over 89 years is replaced with value 91.4. Patients with age equal to zero excluded.

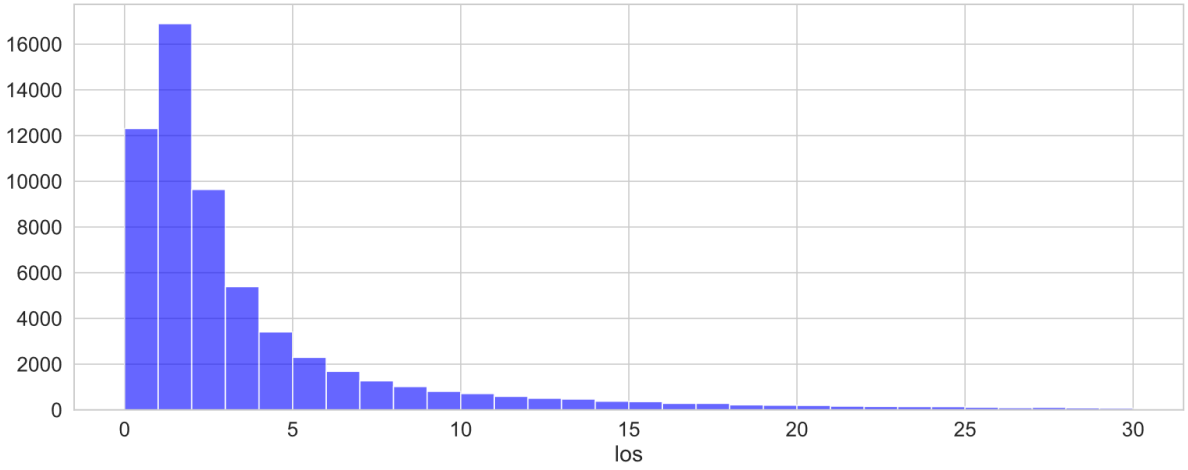

Figure 11 – Histogram of the length of stay. Patients with LOS of over 30 days are not shown.

5.3 Input Variables

Only a small subset of the different measured clinical variables available in the dataset are selected as input variables for the models trained. Selection is based on the variables being commonly measured in the ICU and representing the state of patients' main organ system functions well. The related work reviewed in Chapter 4 also influenced the selection of the input variables. The MIMIC data contains 12 487 distinctly labelled input variables, which, however, also include some duplicate labels for the same concepts as there are different labels for values

measured using CareVue and MetaVision systems. An example of this is that there are measures of heart rate in the database with two different labels, one for each system. In addition to charted input variables, there are 753 labels for different laboratory tests that have been documented to the database. These are equal across the systems used and have the same labels.

The most commonly charted variables include heart rate, respiratory rate, spO2, heart rhythm and different blood pressures. The laboratory exams with most measurements in the database are haematocrit, potassium, sodium, creatinine, chloride, urea nitrogen, bicarbonate, platelet count, anion gap and white blood cells.

The features chosen for model training are selected by reviewing the effectiveness for predicting sepsis in the related studies and by listening to the opinions of subject matter experts in the medical field. They include in total 12 different variables, that represent the patient's vital signs, laboratory tests and demographics. The chosen variables are listed in Table 7.

Table 7 – List of input variables used for training the model.

Vitals Parameters
Heart Rate
Respiratory Rate
Systolic Blood Pressure
Diastolic Blood Pressure
SpO2
Temperature Celsius
Laboratory Parameters
Bilirubin
Creatinine
Glucose
Lactate
Demographics
Age
Gender

5.4 Inclusion Criteria

The original patient population of the MIMIC dataset is narrowed down to form the study cohort using various inclusion criteria. The inclusions are made using similar criteria as in related works by Johnson *et al.* (2018) and Moor *et al.* (2019).

The patients with age under 18 years are excluded from the study cohort. Those patients that had no chart data recorded or missed ICU admission or discharge time were also excluded. The ICU encounters recorded using the CareVue system were left out following examples from recent literature, mainly due to underreported laboratory measurements (Desautels *et al.*, 2016). The ICU stays that are not the patient's first in the hospital are removed, to mitigate the risk of repeated measures. Patients that have their first service documented as cardiothoracic surgery, are excluded as their post-operative physiologic derangements are different from the rest of the ICU population. Lastly, if the suspected infection has appeared too early, the patients are also excluded. (Johnson *et al.*, 2018)

A patient is determined as septic if he or she encounters sepsis onset during any time of his or her ICU stay. The septic patients are the (positive) cases of the cohort. All the patients that do not develop sepsis are the controls. The controls might have had a suspected infection or organ dysfunction as measured by SOFA score change, but they must have appeared separately to not fulfil the sepsis-3 target definition. At this point, each case gets randomly assigned an equal number of control cases, roughly ten controls per one case. To have a sufficient amount of data for the data window, the patients that have their sepsis onset event within 7 hours from admission time or stay in the ICU for less than 7 hours are excluded from the study cohort. This exclusion removes proportionally more cases than control patients and skews the class imbalance. To preserve the prevalence of positive cases close to the original ratio, a case-control matching is performed similarly to Futoma *et al.* (2017) and Moor *et al.* (2019), and controls without assigned case during the matching are excluded from the study cohort. The process for the inclusion criteria can be examined from Figure 12.

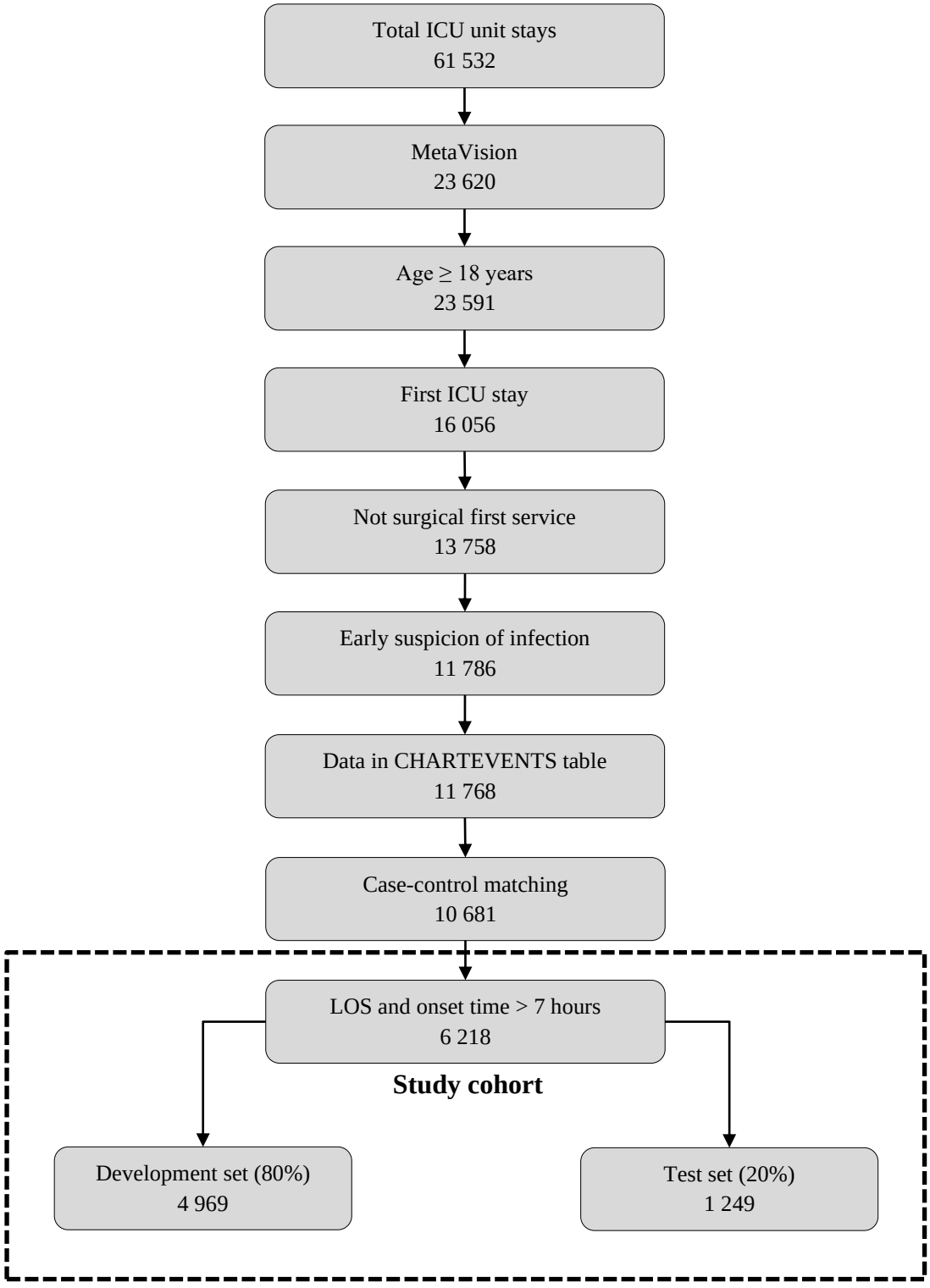

Figure 12 – The applied inclusion criteria diagram. The remaining number of patients is listed in the bottom of each block.

5.5 Study Cohort

The study cohort formed by applying the inclusion criteria, as described in Chapter 5.4, on the original dataset includes 6218 unique ICU stays. 569 of these ICU stays contain a positive sepsis-3 case. The cohort is further divided into two subsets, a development and testing set, so that the training set contains randomly selected 80% of the patients from the original dataset, while the rest are assigned to the testing set. This results into a development set with 4969 patients and a test set with 1249 patients. The patient demographics of the study cohort can be examined from Table 8. The distribution between different admission types for the study cohort patients are shown in Table 9.

Table 8 – Patient demographics in the study cohort.

	Development set			**Test set**		
Demographic	**all (%)**	**pos (%)**	**neg (%)**	**all (%)**	**pos (%)**	**neg (%)**
Total	4969	455	4514	1249	114	1135
Gender						
Male	2773 (55.8)	261 (57.4)	2512 (55.6)	717 (57.4)	71 (62.3)	646 (56.9)
Female	2196 (44.2)	194 (42.6)	2002 (44.4)	532 (42.6)	43 (37.7)	489 (43.1)

Table 9 – Admission types of the patients in the study cohort.

	Development set			**Test set**		
Admission type	**all**	**pos**	**neg**	**all**	**pos**	**neg**
Emergency	4199	401	3798	1093	102	991
Elective	716	49	667	148	11	137
Urgent	54	5	49	8	1	7

The final study cohort includes 10.2% of the ICU stays from the MIMIC dataset. Most of the patients were excluded due to them being documented using the CareVue system. The prevalence of positive cases in the development and test set is 9.2% and 9.1%, respectively. In Table 10, the distribution of age and length of stay are shown for the patients in the study cohort.

Table 10 – Distribution statistics of age and length of stay in the study cohort.

	Development set			Test set		
Demographic	**all**	**pos**	**neg**	**all**	**pos**	**neg**
Age (years)						
Mean	64.2	67.2	63.9	64.2	67.7	63.8
Median	65.4	67.8	65.2	65.0	66.9	64.7
Standard deviation	17.4	15.3	17.6	17.1	14.0	17.3
Length of stay (days)						
Mean	3.4	7.7	3.0	4.0	7.7	3.6
Median	1.9	5.0	1.8	2.1	5.3	2.0
Standard deviation	4.6	7.7	3.9	6.1	7.6	5.8

A preliminary comparison of age and LOS distribution between the positive and negative samples is possible from Table 10. The patients that are labelled positive are older than those that are labelled negative in the study cohort. The LOS of the septic patients is also clearly longer than of those, that do not develop sepsis during their ICU stay. The statistics for the development set and test set seem to be representative of each other with no large deviations for any of the variables.

6 METHODS

This chapter describes the methods used in the experimental part of the study. In Section 6.1, the prediction task is described. Section 6.2 describes several data pre-processing steps undertaken before passing it to the algorithm. In Section 6.3, ways of handling class imbalance are discussed.

Finally, Section 6.4 introduces the models used for the prediction task. It begins with introducing the methods on how the input data, which was common for all the models, was constructed. The process of hyperparameter tuning is briefly discussed. Later, each of the models and their settings are discussed in their own subchapters.

6.1 Prediction Task

The objective of this study is to predict sepsis onsets in intensive care patients. The prediction is done using several recorded measurements of vital signs, laboratory tests and information about the patients' demographics. Three windows need to be chosen to make the predictions: a data window, a gap and a prediction window. The first, data window, determines how and what part of the time-dependent input data is used for the ML model. The larger the data window, the more information about the behaviour of input variables the model can learn but also the complexity of the model increases proportionally to the data window size as more input parameters get gathered. In this study a six-hour input data window is used for all the variables. Large data window would also require patients to have stayed for a longer time in the ICU, before the model can be used.

The prediction window determines how far ahead the algorithm is trying to make predictions and essentially tells the time period during which it estimates that a positive event is or is not going to happen. Lastly, a gap window is simply the time window between the data window and the prediction window. A larger gap window means making predictions earlier. The target class of a data window is determined by examining if an event happens during the prediction window or not, meaning it gets a positive label if an event does occur in the prediction window and negative otherwise:

$$y_t = \begin{cases} 1, & \text{if } t_{\text{time\_of\_sepsis}} \in [t + \text{l}_{\text{gap}}, t + \text{l}_{\text{gap}} + \text{l}_{\text{pred}}] \\ 0, & \text{otherwise} \end{cases} \quad (33)$$

where y_t is the label of a data window ending at time t. The sepsis onset time as defined in Chapter 2.4 is represented by $t_{\text{time\_of\_sepsis}}$. The length of the gap (l_{gap}) and the length of prediction window (l_{pred}) determine the prediction window to which the onset time is compared. The label y_t is positive in case the sepsis onset time ($t_{\text{time\_of\_sepsis}}$) occurs within the prediction window and negative otherwise. In this work, a prediction window of one hour is used. Several different gap lengths from zero to six hours are used and the models' performance is evaluated as a function of the gap window length. Examples of different window length combinations are illustrated in Figure 13.

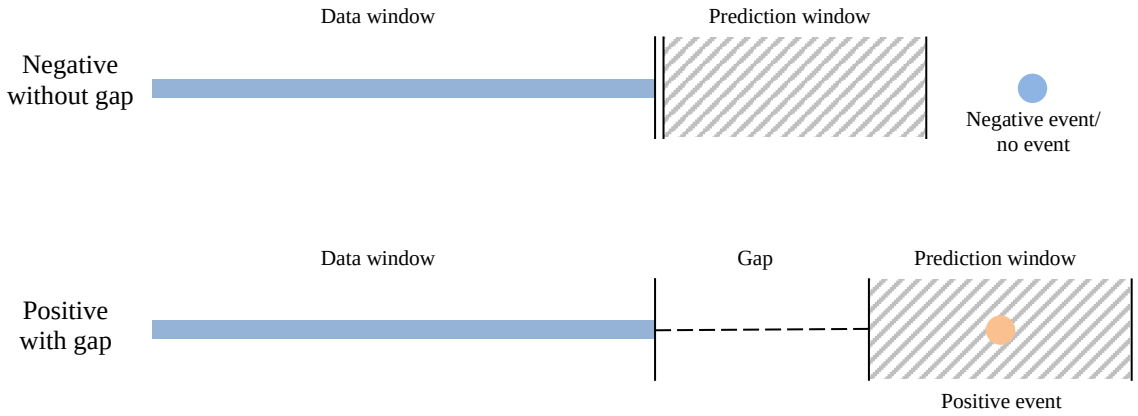

Figure 13 – An illustration of a prediction task. The first example presents a case, where the gap window size is zero and there is no positive event in a prediction window. The second example uses a gap window and has a positive event taking place in the prediction window.

6.2 Data Pre-Processing

The EHR data is pre-processed to make it usable as input data to the ML models. First, filtering is performed to reduce outliers that are clinically abnormal. Next, normalization of data is performed to scale the different variables to similar range for the LSTM model. The tree-based models are not affected by any monotonic transformation as the order of the samples remains

the same, and thus normalization is not applied for them. Following, imputation of missing data is performed. Finally, an hourly matrix of aggregated features from the input data is generated.

6.2.1 Filtering

All the input parameters representing patients' vital signs are range filtered using clinically normal values as the filter's limits. These values have been selected together with experts of clinical domain to exclude measurements in the input data that are very likely to be false measurements. The models are meant to make predictions based on the clinical state of the patients and the input variables are meant to represent that state. An artefact value that is impossibly low or high does likely not represent a state of the patient but some measurement error. To prevent the model from learning any reasons behind outliers, those values are filtered out. A list of limit values for the range filter are listed in Table 11.

Table 11 – Limit values for input parameter range filtering.

Variable	**Lower limit**	**Upper limit**
Heart Rate	20	250
Respiratory Rate	0	100
Systolic Blood Pressure	10	300
Diastolic Blood Pressure	10	300
SpO2	25	100
Temperature Celsius	10	44

An assumption is made that laboratory measurements and patient demographic information are less prone to measurement errors and thus are not subject to any filtering. However, as described in Chapter 5.2, the patients with age over 89 have their age anonymized. For these patients, the age is replaced with value 91.4.

6.2.2 Normalization

The chosen input variables appear within vastly different ranges in the original dataset. For some models this bears challenges in learning from the input data. A solution to tackle the challenge is to scale the input variables so that the distributions of all the features would be in a similar range. (Chollet, 2017)

In this study a technique called z-normalization, also often called standardization, is used. Each variable is normalized independently using the formula:

$$z = \frac{x - \mu}{\sigma} , \tag{34}$$

where z is the normalized value, x is the original value, μ is the mean of the input variable in the population and σ is the standard deviation of the input variable in the population. This way, each value is mapped to the distance of original value from the mean, divided by the standard deviation in the population. If a value is less than the mean value of the population, the mapped value is negative. After the normalization the mean value of each of the input variables is zero and the standard deviation is one. (Kreyszig, 2011)

6.2.3 Missing Data Imputation

The models in this thesis use hourly aggregated values from the data as input variables as described further in Chapter 6.4. However, some input variables are measured less frequently leading to some hourly timeslots with no values appearing. The ML models require the input vectors to be numerical and so the missing values need to be processed.

As the input variables are time series, one commonly used way to process the missing values is forward fill imputation. Using forward fill imputation, a missing value gets assigned the previous non-missing value that has occurred. This value gets imputed as far forward until a new non-missing value is recorded. One problem with using forward fill imputation are the missing values that occur in the beginning of the patient's stay. There no previous values have

occurred, so the logic doesn't work on imputing them. In these cases, the median value of the recorded values for the input variable in the patient population is imputed. An illustration of forward fill imputation is represented in Figure 14.

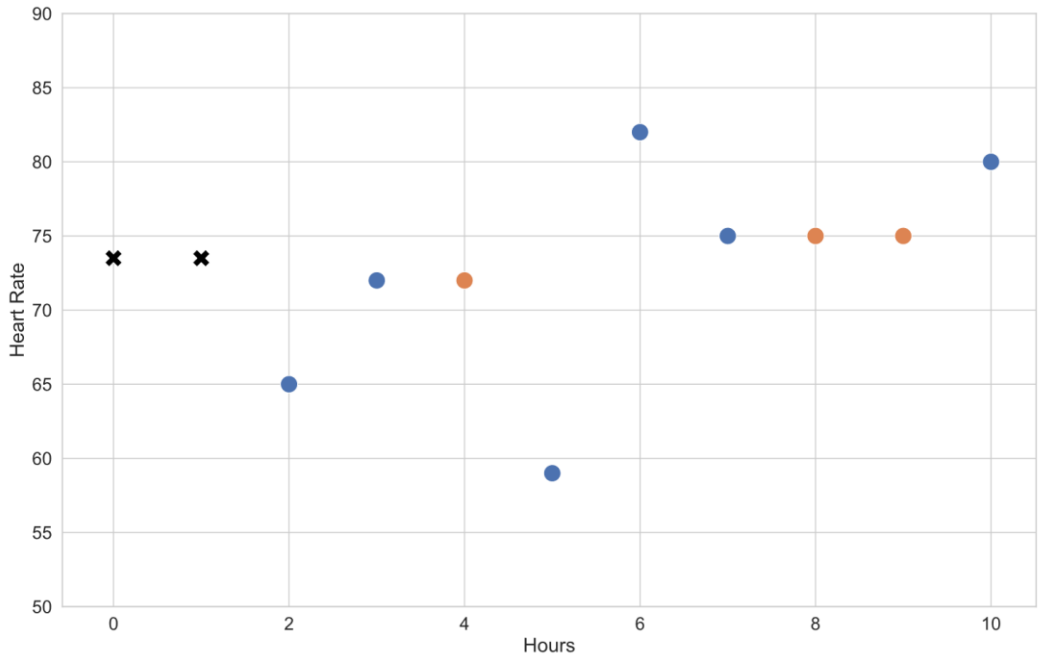

Figure 14 – An illustration of the forward fill method for imputing missing data. The blue dots represent the measurements of heart rate for a patient. Orange dots represent the forward filled missing value of previously occurred measurement, a blue dot. Black crosses represent the median value of the measured blue dots to impute missing values in the beginning of the patient stay.

Additionally for the LSTM model, a binary masking vector is created for each input variable to preserve the information about the missingness of a measurement also after they have been imputed. The masking where $m_t^d \in \{0,1\}^D$ is the masking vector is determined by

$$m_t^d = \begin{cases} 1, & if\ x_t^d\ is\ observed \\ 0, & otherwise \end{cases} . \tag{35}$$

The masking vectors have value zero for each hourly slot that is missing, and value one where a value is present. The vectors are fed as additional input parameters to the trained models that in their term will decide if the information of missingness is important for predicting the target outcomes. (Che *et al.*, 2018)

6.3 Class Imbalance

As often in medical domain, this thesis faces a problem with a low prevalence of the primary outcome. With prevalence of septic patients in the study cohort being roughly 9% as shown in Table 8, the dataset is quite highly unbalanced between target classes. As not all ML models are well equipped to handle and perform well on unbalanced data, an additional technique for balancing is needed to discourage the models from learning to predict the majority class too often. (He and Garcia, 2009)

In this thesis, class weights on misclassification errors are used. They ensure that the models learn both classes equally even when they are facing more examples of the majority class during the training process. The class weights can be obtained by

$$N_o w_o = N_1 w_1 = \frac{N}{2} \quad \rightarrow \quad w_o = \frac{N}{2N_o}, w_1 = \frac{N}{2N_1}\ , \tag{36}$$

where N is the total number of samples in the cohort and N_0, N_1 are the number of samples belonging to negative and positive target classes, respectively. (He and Garcia, 2009)

6.4 Models

Four different model types are trained, and their performance compared in this thesis. The models are random forest, two different gradient tree boosting models and LSTM. The theory for these models has been discussed in Chapter 3.

The measurements of chosen input variables, represented in Table 7, have often been measured at irregular time-intervals in the MIMIC dataset. For some patients a variable has been measured many times an hour and for others there could be only few measurements in the span of several hours. The chosen ML models cannot handle input vectors of different dimensions meaning the data needs to be aggregated first. A common approach for aggregating EHR data for ML tasks is to calculate representations of patients' hourly state that this thesis work also uses. Several different aggregations calculated for recorded clinical values and lab

measurements can be seen in Table 12. No aggregations were needed for patient demographics as they are assumed to stay constant throughout the patients' ICU stays.

Table 12 – List of aggregated features from the input variables.

Vitals Parameters	**Aggregated features**
Heart Rate	Mean, Median, Min, Max, First, Last
Respiratory Rate	Mean, Median, Min, Max, First, Last
Systolic Blood Pressure	Mean, Median, Min, Max, First, Last
Diastolic Blood Pressure	Mean, Median, Min, Max, First, Last
SpO2	Mean, Median, Min, Max, First, Last
Temperature Celsius	Mean, Median, Min, Max, First, Last
Laboratory Parameters	
Bilirubin	Mean, Median, Min, Max
Creatinine	Mean, Median, Min, Max
Glucose	Mean, Median, Min, Max
Lactate	Mean, Median, Min, Max

In total, there are 54 different input parameters available for each hour, when the age and gender of a patient are combined with the aggregated features. For the LSTM model, a binary masking vector indicating missing records within the hourly slot is added for each variable, bringing the total to 64 hourly input parameters. The hourly aggregation of values is illustrated in example Figure 15. After the hourly aggregation has been performed, the next step is to use forward fill imputation.

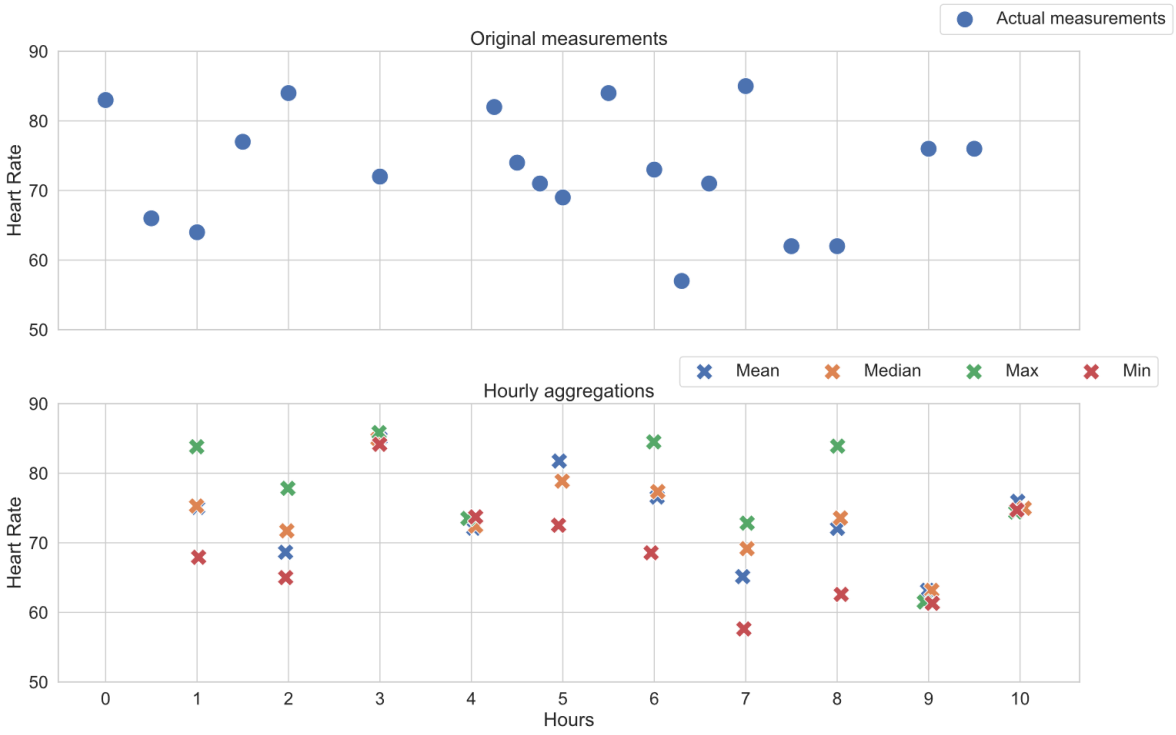

Figure 15 – Illustrations of four different hourly aggregations for a single patient. In the upper plot, the original measurements are represented as blue dots. In the lower plot, the different colored crosses represent aggregated values of the previous hour. Slight jitter has been applied to the markers in the lower plot to make such values visible that have equal values between two aggregations. The plots share the x-axis.

The hyperparameters of each model are optimized using grid search. For each parameter combination, a 5-fold cross-validation is performed on the development set. For all models, the cross-validation is repeated randomly three times with the exception of LSTM in case of which it is only done using one repetition. The parameter combinations for final models are selected based on the mean AUROC value for the cross-validation folds. The parameter combination with the highest mean AUROC value is selected for each model.

6.4.1 Random Forest

Random forest is a commonly used machine learning model. The theory behind the model was discussed in Chapter 3.6. The implementation used for the random forest model in this thesis is from the Scikit-learn Python package.

A grid search for hyperparameters for the random forest model was performed on the number of trees in the forest, the maximum tree depth, minimum number of samples to split an internal node and minimum number of samples to be at a leaf node. The different values for each hyperparameter can be seen in Table 13, from which 81 different parameter combinations are obtained and tested on the development set.

Table 13 – The tested hyperparameters for the Random Forest model.

Hyperparameter	**Tested values**
Number of trees in the forest	500, 1000, 1500
Maximum tree depth	5, 10, 15
Minimum number of samples to split an internal node	2, 5, 10
Minimum number of samples to be at a leaf node	1, 2, 4

6.4.2 Gradient Tree Boosting

A gradient tree boosting is an ensemble model based on the gradient boosting method using decision trees as weak learners. The theory of gradient tree boosting is discussed in Chapter 3.6. In this thesis, two different gradient boosting models, one by Scikit-learn and one by XGBoost, are used.

XGBoost, which stands for “Extreme Gradient Boosting”, implements a few additions to the regular gradient boosting method. It uses training on a subset of samples in the training set, regularization, and column subset sampling on splits, where a randomly selected subgroup of input variables is searched similarly to the random forest algorithm.

The hyperparameters tuned for both the standard gradient boosting and the XGBoost models are shown in Table 14 and Table 15, respectively. The values were selected around the default values for the models and, thus, differ between the two. For the standard gradient boosting model, number of boosting stages, maximum depth, minimum number of samples to split an internal node and minimum number of samples to be at a leaf node were selected for the grid search. For XGBoost, the learning rate, maximum depth, minimum child weight and the gamma parameter were tuned.

Table 14 – The tested hyperparameters for standard Gradient Boosting model.

Hyperparameter	Tested values
Number of boosting stages	50, 250, 500
Maximum depth	3, 5, 8
Minimum number of samples to split an internal node	2, 5, 10
Minimum number of samples to be at a leaf node	1, 2, 4

Table 15 – The tested hyperparameters for XGBoost model.

Hyperparameter	Tested values
Learning rate	0.05, 0.1, 0.15
Maximum depth	3, 4, 5
Minimum child weight	1, 3, 5
Gamma	0, 0.1, 0.2

6.4.3 LSTM

The only deep learning model examined in this thesis work is LSTM. The theory behind recurrent neural networks and LSTM is introduced in Chapter 3.8.2. The LSTM model used is by Keras API using TensorFlow backend.

The hyperparameters are not the only factor in the LSTM model's performance as the neural network's shape can be chosen quite arbitrarily. To factor both the model shape and the hyperparameters, the tuning of the LSTM model was performed in two phases as a complete grid search using both would become computationally too expensive. First, different model shapes including varying number of layers and units per layer are examined using the LSTM model's default parameters. Second, the hyperparameters are tuned for the model shape that performed the best in the first phase. The tested model shapes can be seen in Table 16 and tested hyperparameters in Table 17.

Table 16 – Tested model shapes for LSTM model.

Shape parameter	Tested values
Number of layers	1, 2
Units per layer	4, 8, 16, 32, 64, 128, 256, 512

Table 17 – Tested hyperparameters for LSTM model.

Hyperparameter	Tested values
Learning rate	0.0001, 0.001, 0.01
Batch size	4, 8, 16, 32, 64, 128, 256

Due to long training time and a large number of different parameter combinations the cross-validation is done using 20 epochs. The final model is trained for 50 epochs. Using a smaller number of epochs during the cross-validation gives the initial results about the best parameter combinations in reasonable time, while the larger number of epochs during the training of the final model gives it more time to optimize and possibly perform better on the testing set.

7 EXPERIMENTS AND RESULTS

In this chapter, the experiments performed for the thesis work and their results are examined. First, the results of cross-validation phase are examined with respect to the tuned hyperparameters. Secondly, each model's performance on the test set is examined.

7.1 Cross-Validation Results

For each model type, the model and the hyperparameter combination with the highest mean AUROC score was chosen as the best of its type. Overall, the best cross-validation performance was achieved by a regular gradient tree boosting model in terms of AUROC. The highest cross-validation AUPRC was achieved by an XGBoost model. The confidence intervals were similar across model types, except for LSTM having larger AUROC confidence interval than the rest of the models. The cross-validation performance results can be seen in more detail in Table 18.

Table 18 – The best models' cross-validation performances at sepsis onset. The best model for each model type was chosen by the highest mean AUROC score. (CI = Confidence Interval)

Model	**Mean AUROC** (95% CI)	**Mean AUPRC** (95% CI)
Random Forest	0.898 (± 0.008)	0.617 (± 0.023)
Gradient Boosting	0.920 (± 0.005)	0.641 (± 0.023)
XGBoost	0.919 (± 0.006)	0.650 (± 0.025)
LSTM	0.878 (± 0.024)	0.513 (± 0.031)

The best performing random forest model used 1500 estimators, maximum tree depth of 15, minimum number of samples to split an internal node of 10 and minimum number of samples to be at leaf node of 4. The mean of the mean AUROCs of the random forest models resulted from the hyperparameter tuning are plotted as a function of the hyperparameter in the Figure 16. The selected hyperparameters performed best as a combination, and some hyperparameters can be seen to achieve higher mean AUROC values across all the other hyperparameter combinations when examined separately.

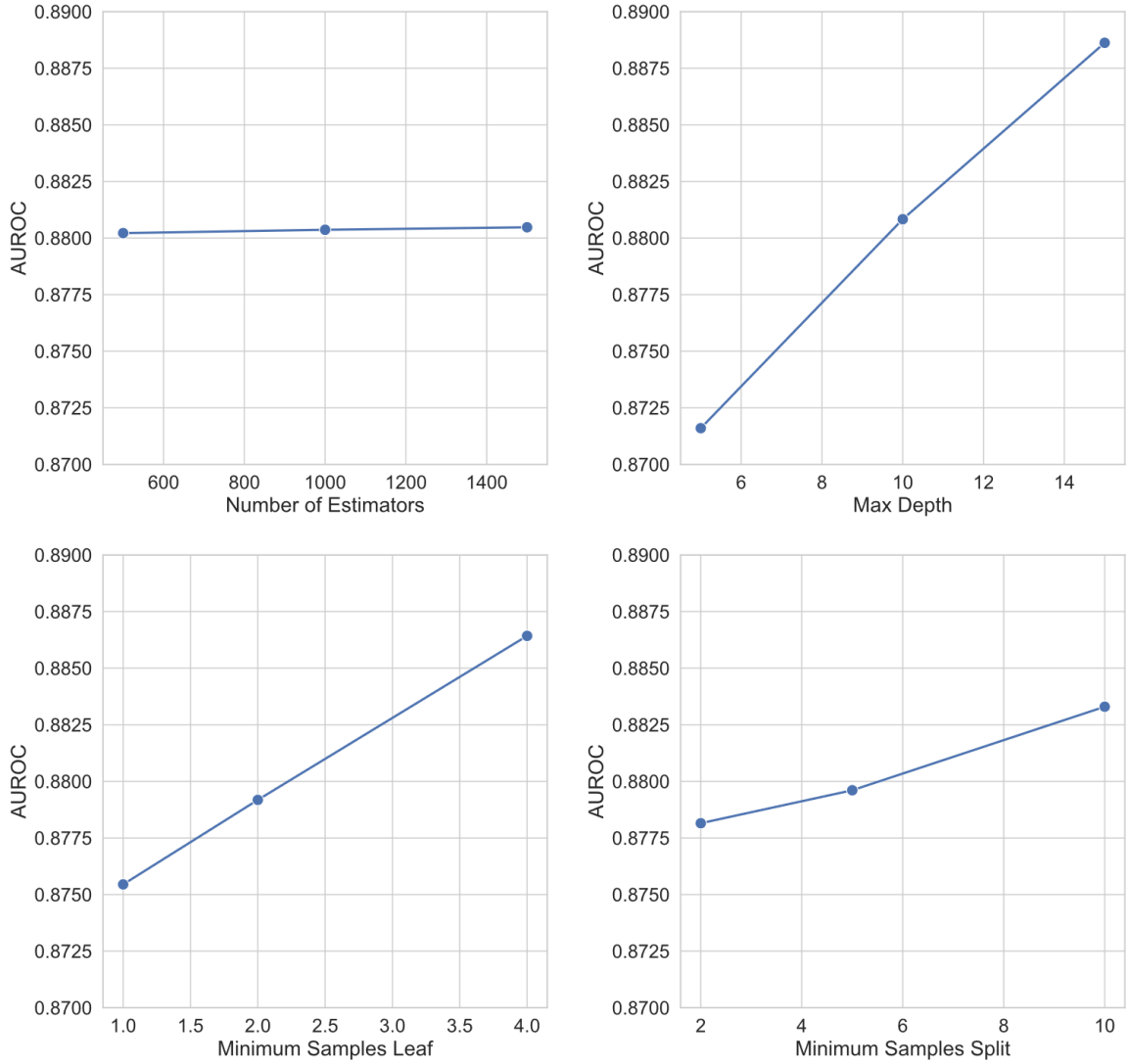

Figure 16 – Cross-validation results for the random forest model. Each subplot represents the mean AUROC as the function of a hyperparameter.

Of the tested regular gradient tree boosting models, the best results in terms of AUROC were achieved with one that used 500 boosting stages, had maximum depth of 8, minimum number of samples to split an internal node of 10, and minimum number of samples to be at leaf node of 1. The mean AUROCs of regular gradient tree models during the hyperparameter tuning are plotted as a function of the hyperparameter in Figure 17.

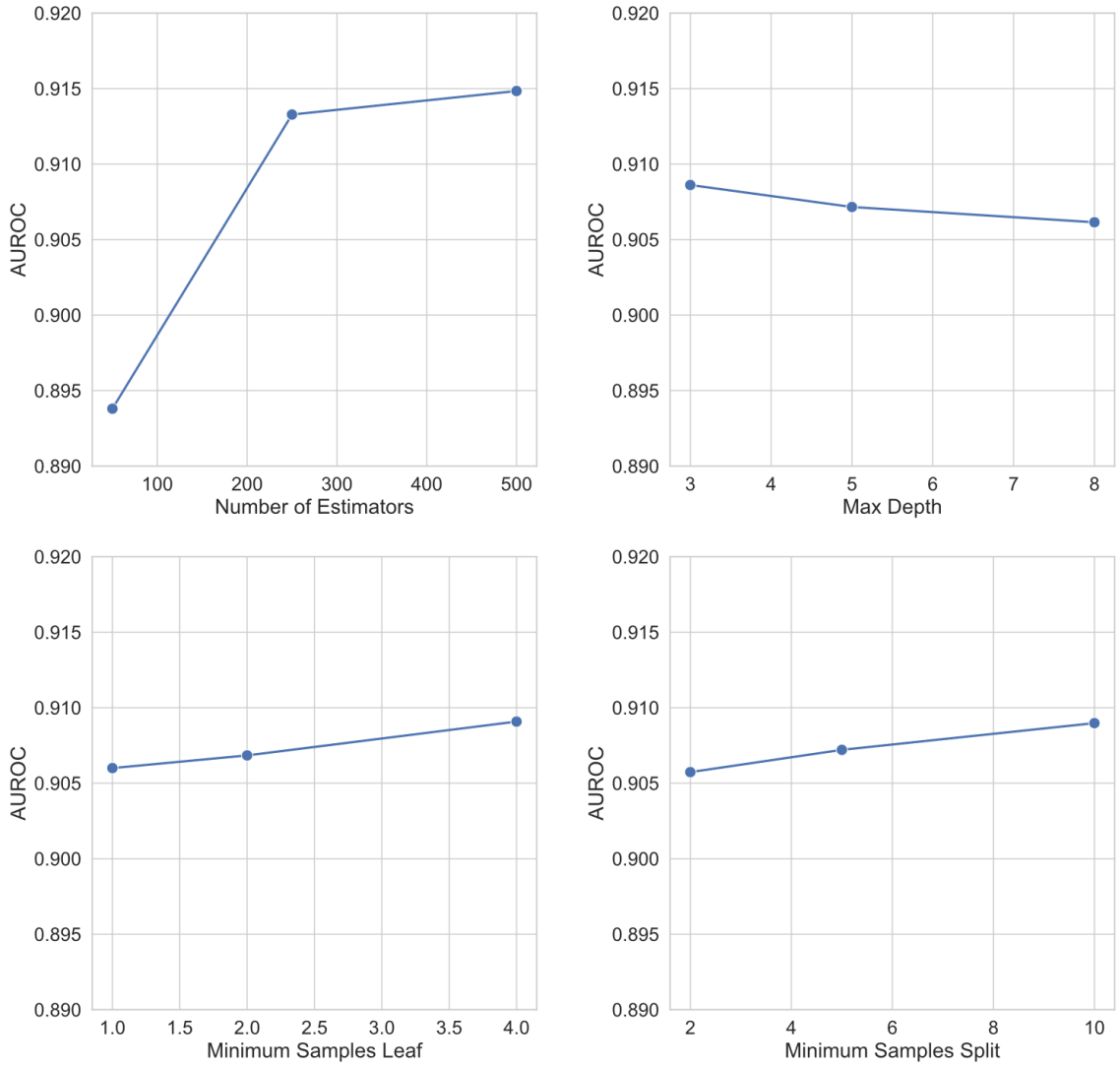

Figure 17 – Cross-validation results for the gradient boosting model. Each subplot represents the mean AUROC as the function of a hyperparameter.

The XGBoost model that achieved the best cross-validation mean AUROC results had learning rate of 0.1, maximum depth of 4, minimum child weight of 5 and gamma of 0.1. The mean of the mean AUROCs of the XGBoost models resulted from the hyperparameter tuning are plotted as a function of the hyperparameter in the Figure 18.

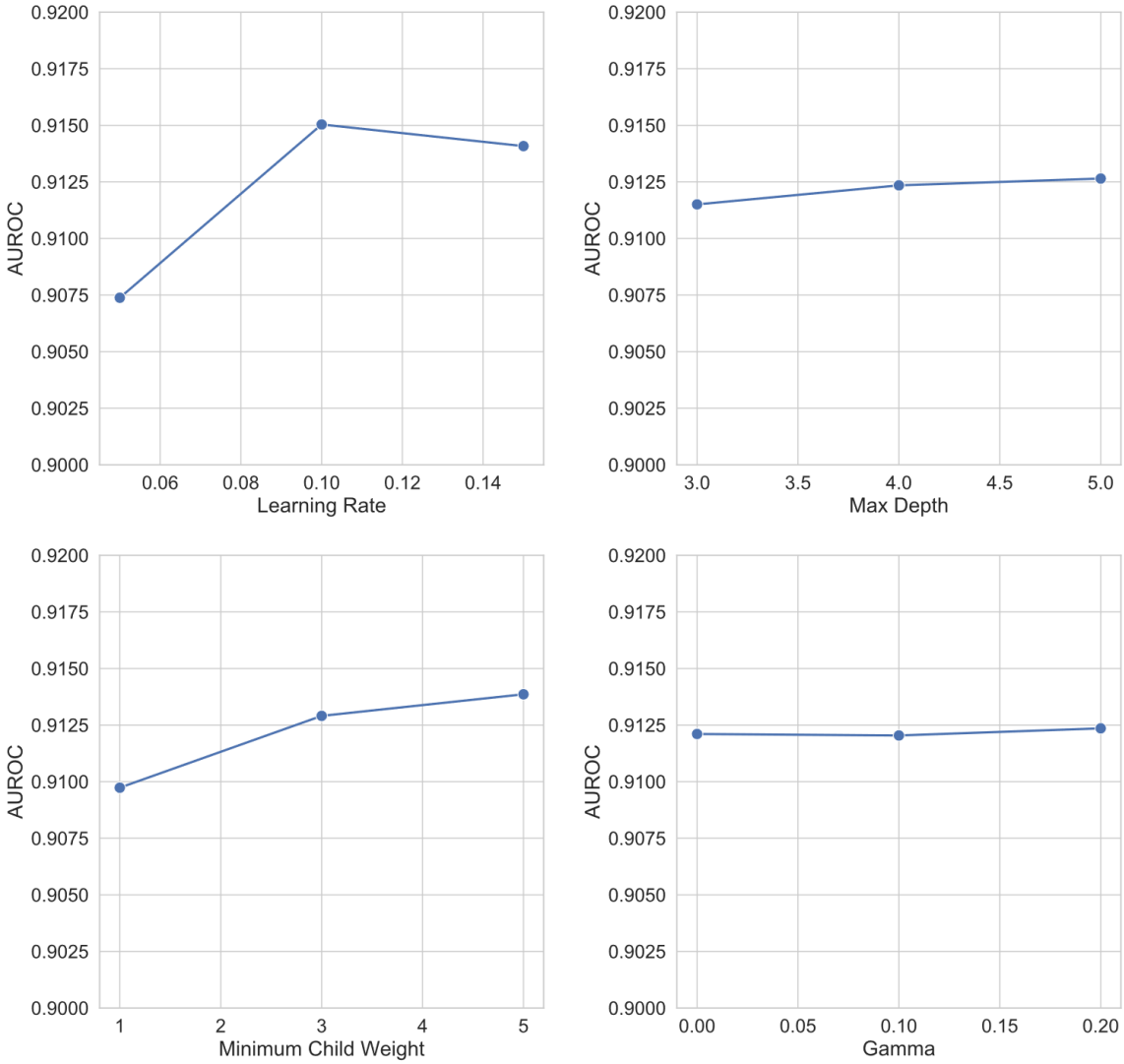

Figure 18 – Cross-validation results for the XGBoost model. Each subplot represents the mean AUROC as the function of a hyperparameter.

The LSTM model was tuned in two phases. First, the model structure was optimized using grid search from parameters in Table 16. The best performing model was one with a single layer and four units in the layer. In the next phase, the learning rate and the batch size of the model were optimized using grid search. The best performing model in terms of mean cross-validation AUROC had learning rate of 0.001 and batch size of 4. The mean of the mean AUROCs of the LSTM models resulted from the hyperparameter tuning are plotted as a function of the hyperparameter in Figure 19.

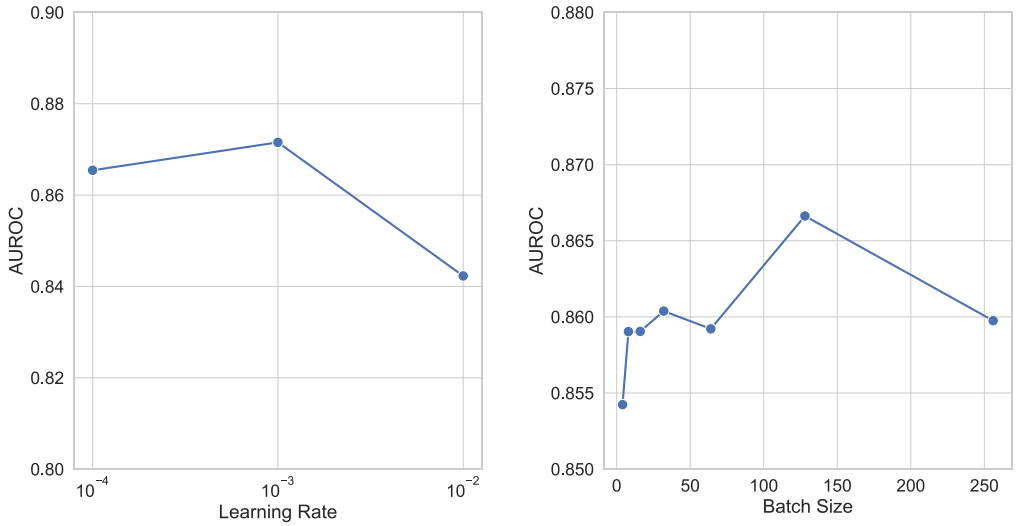

Figure 19 – Cross-validation results for the LSTM model. Each subplot represents the mean AUROC as the function of a hyperparameter.

7.2 Test Results

The hyperparameters for each model type that performed the best in terms of the mean AUROC during the cross-validation were chosen for developing the final models. The final models use the whole development set as their training data and the test set to evaluate the models' performance. For each model type four separate models with different prediction gaps were trained and evaluated. The gaps, or offsets from the sepsis onset time, were 0, 1, 3 and 6 hours. With larger gap from the sepsis onset, the models were fed with data measured earlier and further away from the onset time leading to a model that is trained to make the predictions further into the future. The different offset combinations are illustrated in Figure 20.

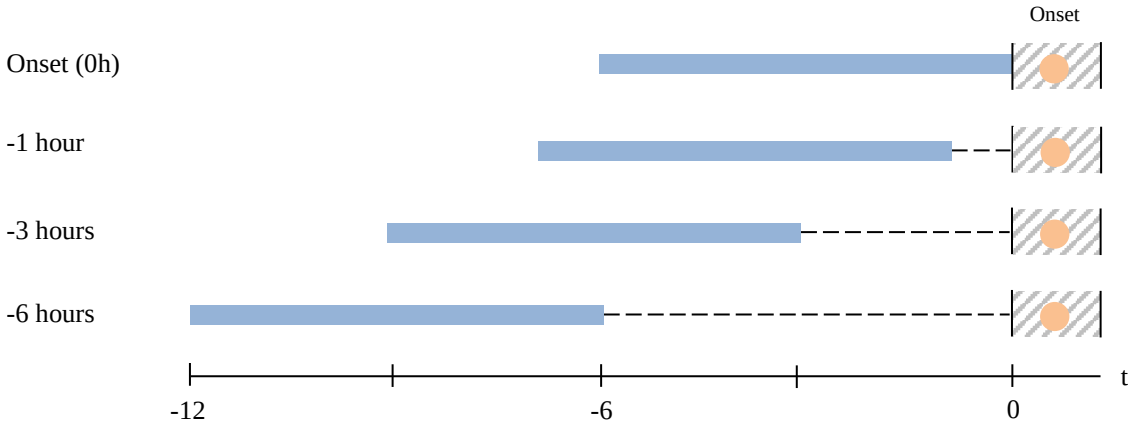

Figure 20 – Illustration of the different prediction times. The blue bar represents the data window. The dashed line represents the gap. The box with filled line pattern is the prediction window. The orange dot represents the onset time.

The best performance on the test set was achieved by the XGBoost model, which got both the best AUROC and AUPRC scores on most of the tested prediction gaps. The AUROC and AUPRC plots of the models' performance at onset can seen be from Figure 21. When comparing the models' performance by binary metrics, the sensitivity was fixed to make the rest of the statistics comparable, as they are threshold dependent. The binary metrics comparison was done on the models making prediction at onset time. The sensitivities used were 0.25, 0.5 and 0.75, and the results for them are in Table 19, Table 20 and Table 21, respectively. Also by the binary metrics, the XGBoost outperformed the rest of the models in most cases, with regular gradient tree boosting model being better in comparison with sensitivity of 0.25.

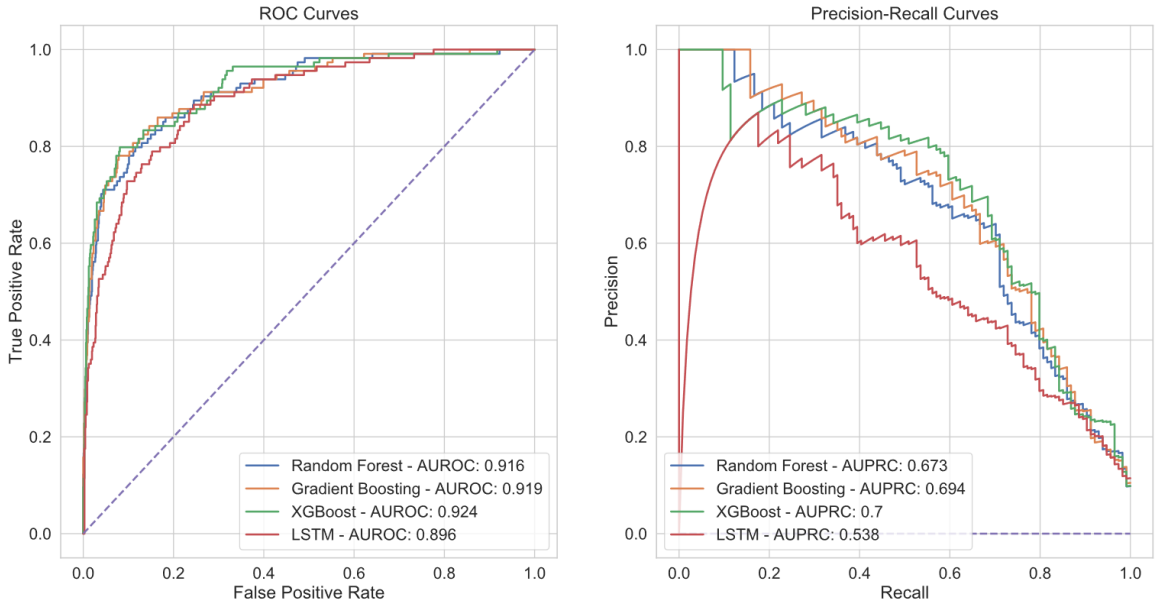

Figure 21 – Model performance at onset. The left plot shows the Receiver Operating Characteristic curves and the right plot the Precision-Recall curves of the models.

Table 19 – The binary performance statistics for models predicting at onset with fixed sensitivity of 0.25.

Metric	**Random Forest**	**Gradient Boosting**	**XGBoost**	**LSTM**
Threshold	0.808	0.986	0.938	0.385
Accuracy	0.926	**0.929**	0.927	0.924
TPR (sensitivity)	0.250	0.250	0.250	0.250
TNR (specificity)	0.995	**0.997**	0.996	0.992
FPR	0.005	**0.003**	0.004	0.008
FNR	0.750	0.750	0.750	0.750
PPV (precision)	0.824	**0.903**	0.871	0.757
NPV	**0.929**	**0.929**	**0.929**	**0.929**
LR+	46.462	**92.924**	67.204	30.975
LR-	0.758	**0.756**	0.766	0.760
F1-score	0.378	**0.386**	0.372	0.371

Table 20 – The binary performance statistics for models predicting at onset with fixed sensitivity of 0.5.

Metric	Random Forest	Gradient Boosting	XGBoost	LSTM
Threshold	0.606	0.634	0.831	0.256
Accuracy	0.937	0.942	**0.944**	0.923
TPR (sensitivity)	0.500	0.500	0.500	0.500
TNR (specificity)	0.981	0.986	**0.989**	0.967
FPR	0.019	0.014	**0.011**	0.033
FNR	0.500	0.500	0.500	0.500
PPV (precision)	0.722	0.781	**0.814**	0.596
NPV	0.951	**0.952**	**0.952**	0.950
LR+	25.795	35.469	**43.654**	14.672
LR-	0.510	0.507	**0.506**	0.526
F1-score	0.591	0.610	**0.620**	0.538

Table 21 – The binary performance statistics for models predicting at onset with fixed sensitivity of 0.75.

Metric	Random Forest	Gradient Boosting	XGBoost	LSTM
Threshold	0.343	0.023	0.516	0.152
Accuracy	0.890	0.910	**0.914**	0.861
TPR (sensitivity)	0.750	0.750	0.750	0.750
TNR (specificity)	0.903	0.926	**0.930**	0.873
FPR	0.096	0.074	**0.070**	0.127
FNR	0.250	0.250	0.250	0.250
PPV (precision)	0.438	0.503	**0.518**	0.371
NPV	**0.973**	**0.973**	**0.973**	0.972
LR+	7.764	10.075	**10.712**	5.877
LR-	0.281	0.275	**0.274**	0.291
F1-score	0.552	0.601	**0.612**	0.496

The LSTM model seems to suffer less in performance drop when increasing the prediction gap and predicting from further away when compared to the other models. The regular gradient tree boosting and the random forest model both lose more of their predicting power when increasing the gap length. The XGBoost model also drops in the performance more than the LSTM model but it still outperforms all the other models with the six-hour prediction gap. The AUROC and AUPRC performance results with respect to the prediction gap for all the models are plotted in Figure 22 and listed in Table 22.

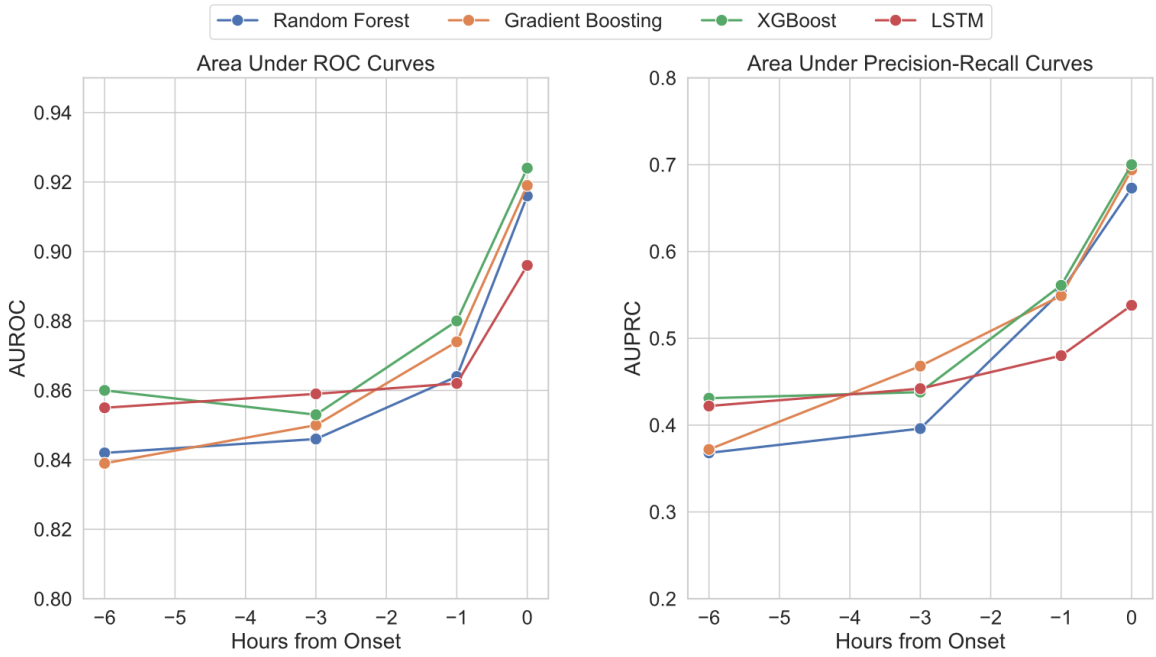

Figure 22 – The models' performance as a function of the prediction gap length. The left plot shows the AUROCs and the right plot the AUPRCs of the models.

Table 22 – The models' test set performances with different prediction gap lengths.

Model	**AUROC**				**AUPRC**			
	0h	**-1h**	**-3h**	**-6h**	**0h**	**-1h**	**-3h**	**-6h**
Random Forest	0.916	0.865	0.846	0.842	0.673	0.554	0.396	0.368
Gradient Boosting	0.919	0.874	0.850	0.839	0.694	0.549	**0.468**	0.372
XGBoost	**0.924**	**0.880**	0.853	**0.860**	**0.700**	**0.561**	0.438	**0.431**
LSTM	0.896	0.862	**0.859**	0.855	0.538	0.480	0.442	0.422

7.3 Clinical Setting and Hourly Predictions

In the real-world clinical setting, a predictive model would need to do continuous predictions on the patients during their whole stay at the ICU. The models trained in this thesis use only a snapshot of the patients' ICU stays, as there is only a single prediction time and a relatively short data window that is fixed to be right before the prediction time.

An additional experiment is made for the XGBoost model to examine the clinical predictive power of a model that was developed using fixed prediction times on the development set against a model that uses data from the whole stays in the development set. The predictions are made hourly, and the task is to predict if sepsis onset will occur during the next three hours using the data from the last six hours. The process is essentially sliding the data and prediction window across the patient stays and is illustrated in Figure 23. For healthy patients, the whole ICU stay is used for the training. In case of septic patients, once the sliding window reaches the occurrence of the true onset in its prediction window, no more predictions are made for the rest of the ICU stay.

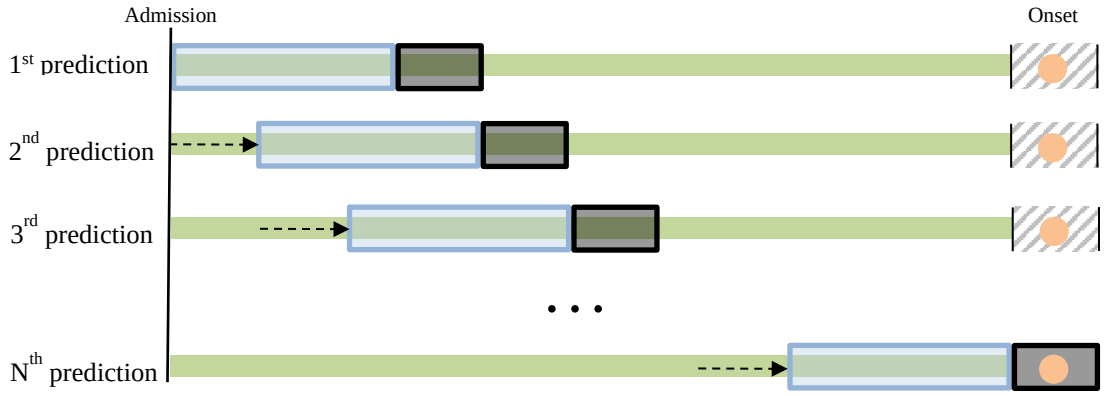

Figure 23 – An illustration of hourly prediction for a septic patient during their whole ICU stay. The blue data window and the black prediction window are slid across the whole length of the stay to achieve hourly predictions.

The performance results of the clinical setting experiment can be addressed in two different ways. First is to examine the predictions datapoint-wise and evaluate if each of the predictions was made correctly or incorrectly. Another approach is to examine the performance patient-wise. Patient-wise performance aims to count patients that were diagnosed as septic at least

once during their stay. The patient-wise predictions are calculated by taking the maximum over the model's predictions for patients' hourly data instances.

There are a total of 83316 prediction datapoints in the test set when using the six-hour input data window and hourly predictions. The first prediction is made at six hours after patient's admission. In datapoint-wise comparison each datapoint gets a prediction from both the continuous and the fixed model. As the patients in the test set are the same as before, there are 1245 patient-wise probabilities calculated for both models.

The model that was trained using the whole patient stays performed considerably better than the one that used fixed predicting times from the development set. The datapoint-wise performance of the continuous model for hourly predictions on the test set was 0.896 in terms of AUROC and 0.086 in terms of AUPRC. The prevalence of positive cases in the datapoint-wise prediction less than 0.4%, making the data extremely unbalanced. The fixed model achieved AUROC of 0.72 and AURPC of 0.016 on the datapoint-wise predictions. The ROC and Precision-Recall curves can be seen from Figure 24.

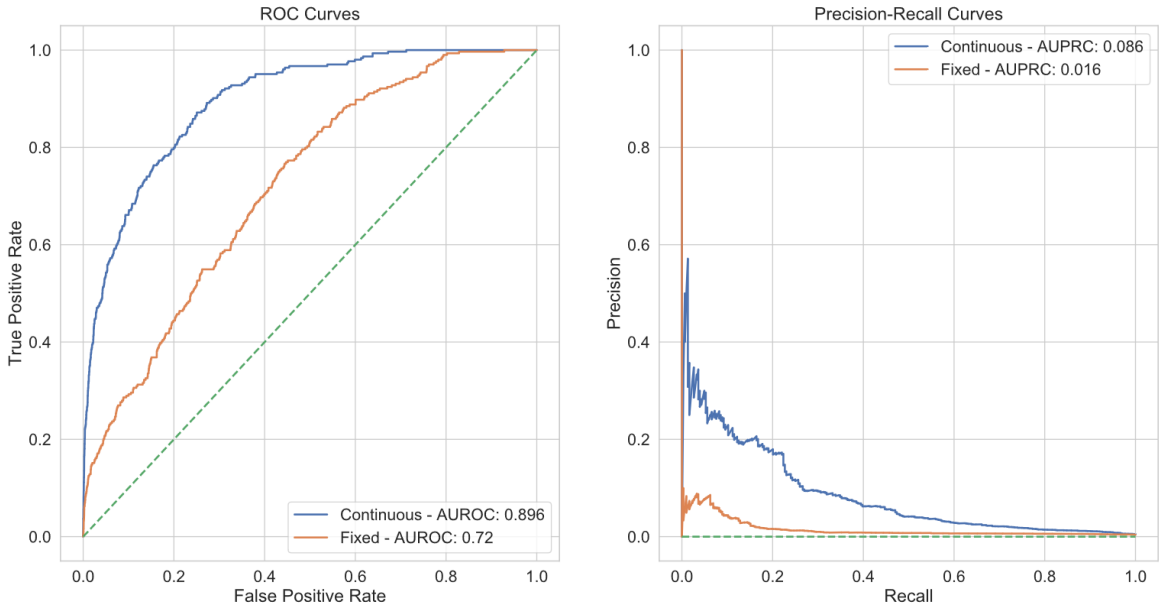

Figure 24 – Datapoint-wise performance of the continuous and fixed models on the test set for hourly predictions during the whole ICU stays.

The continuous model also performed better than the fixed model in patient-wise prediction performance comparison, although the difference between them is smaller than in datapoint-wise performance. The continuous model achieved patient-wise AUROC of 0.801 and AURPC of 0.329. The results for the fixed model were 0.74 for AUROC and 0.177 for AUPRC performance, respectively. The patient-wise prevalence was 9.1%. The patient-wise performance plots are in Figure 25.

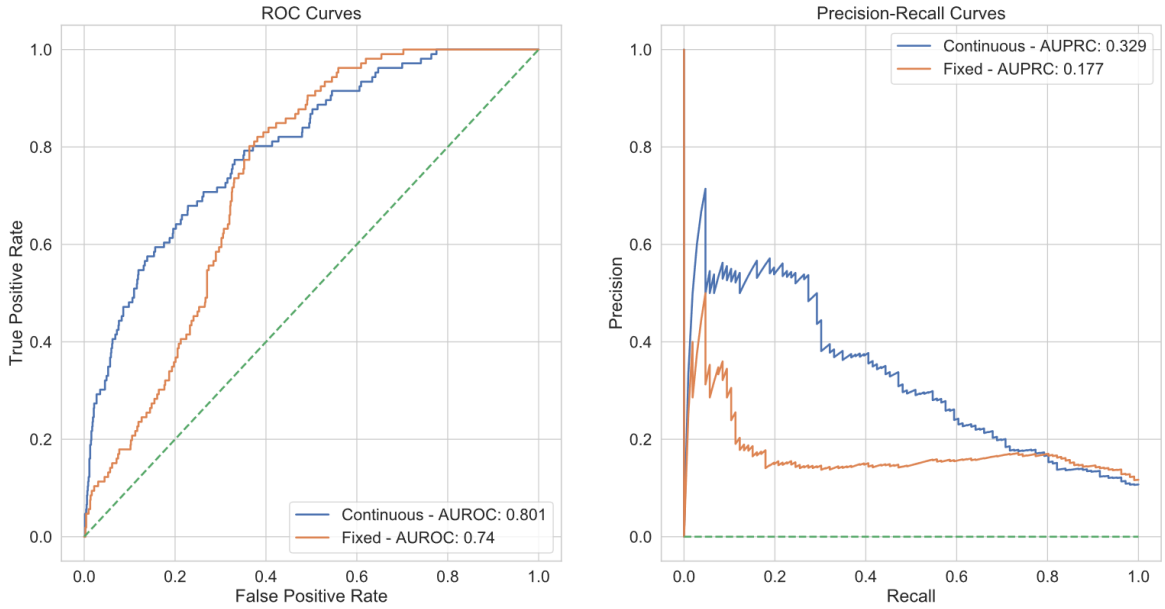

Figure 25 – Patient-wise performance of the continuous and fixed models on the test set for hourly predictions during the whole ICU stays.

8 DISCUSSION

This thesis examines the performance of several machine learning models on predicting sepsis in the intensive care unit. The models' performance is compared by their AUROC and AUPRC scores using different prediction gaps. Additionally, an experiment is made to examine the value of using data from the whole patient stays when using the model in a clinical setting. First, this section includes discussion on the performance of the models. Secondly, the limitations that apply to the study are discussed.

8.1 Performance of the Models

All the developed model types achieved AUROC value in the range of 0.89–0.92 on the test set, when the predictions were made at the sepsis onset time. The AUPRCs scores at the same prediction time were in the range of 0.54–0.7. As the cross-validation and hyperparameter tuning were conducted using the same prediction times, the training and testing performance can be compared. For all the models, the AUROC score remained essentially unchanged between the training and testing and the AUPRC score was higher on the testing set for all the models indicating there should not be much overfitting in the models. These results seem to be similar to those achieved in reviewed related work.

Comparing the models' performance change with larger prediction gaps showed similar behaviour in all the models. The performance was best when predicting closest to the onset. Lowest performance resulted from predicting with the largest tested gap of six hours. However, not all the models suffered equally with the challenge of predicting from further away. The XGBoost and LSTM models show better results using six-hour gap length compared to random forest and regular gradient tree boosting models, while close to onset LSTM is the only model performing clearly worse than the rest.

The XGBoost model's performance in a clinical setting is evaluated using an experiment. Two versions of the model are trained, one using only fixed parts of the patient stays similarly to the other models developed in this thesis, the other using whole patient stays and continuous hourly predictions. The model that was developed using continuous hourly predictions also performed

clearly better on making predictions on the test set that had prevalence of positive cases of 0.36%. The datapoint-wise AUROCS were 0.896 and 0.720 and AUPRCs 0.086 and 0.016 in the continuous and fixed model, respectively. The result indicates that using all the available data and staging the prediction task similarly to how it would be used in a real-world use-case could lead to better performance. The better performance can lead to lives saved and unnecessary costs avoided as the model is able to predict the septic patients better and avoid making false alarms leading to alarm fatigue in the medical staff.

8.2 Limitations

This study contains some limitations. Being a retrospective study conducted on data from a single hospital centre, the results may not be generalized to other locations and clinical settings. The quality of the data cannot be guaranteed in all cases and the measured values can be suspect to errors due to sensor malfunctioning or invalidly entered data by clinicians. This was partly addressed by removing outliers using range filtering. The clinical practices between hospitals can vary, and the model might have captured some of these behaviours when making the prediction. Testing the models' performance on data from other locations would evaluate their generalizability.

The derivation of sepsis onset times used as target labels can be questioned. Even if the used practice follows that of currently available Sepsis-3 definition and methods used in related work, sepsis remains a condition without a definitive diagnostic test. Using different window lengths around the suspected infection time will lead to some cases being labelled differently. The definition of suspected infection time depends on the clinicians' practice on acquiring blood cultures and administrating antibiotics, which may not be the same between clinical settings, may change between different time periods and can also be subject to human errors. The definition of sepsis is likely to change again sometime in the future, and when it does, the results of this thesis may no longer apply as such.

The cohort selection process might have resulted in some bias to the results. The requirement of data being recorded using the MetaVision system and the ICU stay being the first for each patient decreased the cohort size by almost three quarters from what was available in the

original dataset. Discarding patients with LOS of less than seven hours or onset before seven hours from admission reduced the number of septic patients from the original dataset. However, including patients with less data than the selected data window length would likely not lead to meaningful results for the goal of this thesis. The decision to use case-control-matching decreased the number of healthy patients. Using only fixed data windows might capture more such periods where the healthy patients were in a stable condition, as the data windows were selected randomly for non-septic patients. The opposite could also be possible.

The chosen input variables and their hourly aggregation before feeding them to the models might not represent the patients' condition in the best possible way regarding the prediction task. Any other combination of input variables could lead to better performance. The chosen input variables might also include such variables that have bias towards differing clinical practice for sick patients compared to healthy patients. For example, a measurement for some variable might be done more frequently for a patient that the clinical staff suspects is under risk of developing a condition. Using other means of aggregating the input data and filling the missing values could result in better results.

The hyperparameter tuning for the chosen models is performed only with a limited range of values and combination of hyperparameters. The argument for doing a more extensive hyperparameter search for the models is valid and should be done when moving forward with researching the models further. The aim of this thesis was to evaluate the performance of different model types, and with limited computing power as well as time available the obtained results serve as initial suggestions on how different model types might perform for predicting sepsis. The hyperparameter tuning was also performed only for the models predicting at sepsis onset and could be done for every prediction gap separately.

The comparison of a model trained with fixed versus continuous predictions in the development set and their performance in a clinical setting suggest that there is value from using as much data as possible from the patient stays. This experiment should be extended also to the other model types, as using the continuous approach would likely benefit them too. Different data and prediction window lengths could affect the models' feasibility as a sepsis prediction tool in

real-world clinical practice. However, this falls outside the scope of this thesis work and should be conducted in further research.

9 CONCLUSIONS

The aim of this study was to examine what methods can be effectively used for predicting sepsis during the ICU stay. Four different model types: random forest model, two different gradient tree boosting models, and LSTM model were developed, and their predictive performance tested. The hyperparameters for each model were tuned using grid search and cross-validation. The evaluation was done using several different prediction gap lengths. Heart rate, respiratory rate, systolic and diastolic blood pressures, SpO2, temperature, bilirubin, creatinine, glucose, lactate, age and gender were used to make the predictions.

The XGBoost model showed the best results out of the tested models, performing best in most of the tests done. Every model's predictive performance decreased when increasing the prediction gap length. An experiment of using the XGBoost model in a clinical setting and making hourly predictions showed the added value of using all the available patient stay data for developing the model.

This study successfully identified methods that can be used to determine target labels for sepsis onset from the reviewed related work. Models were successfully developed and tested on the MIMIC-III dataset. Future work is required to tune the models and evaluate their performance on a multi-centre study, before drawing conclusions on the clinical applicability. Additionally, using different input variables should be further examined. Especially the benefit of including infection related biomarkers like C-reactive protein and procalcitonin, which were not available or routinely collected in the datasets considered for this thesis work, would be interesting to research.

REFERENCES

Alpaydin, E. (2014) *Introduction to Machine Learning*. 3rd ed. MIT Press.

Ayers, S., Baum, A., Newman, S., Wallston, K., Weinman, J. and West, R. (2007) *Cambridge Handbook of Psychology, Health and Medicine*. 2nd ed. Cambridge University Press. doi: 10.1017/CBO9780511543579.

Barton, C., Chettipally, U., Zhou, Y., Jiang, Z., Lynn-Palevsky, A., Le, S., Calvert, J. and Das, R. (2019) 'Evaluation of a machine learning algorithm for up to 48-hour advance prediction of sepsis using six vital signs', *Computers in Biology and Medicine*, 109, pp. 79–84. doi: 10.1016/j.compbiomed.2019.04.027.

Bishop, C. M. (2006) *Pattern Recognition and Machine Learning*. Springer.

Breiman, L. (2001) 'Random Forests', *Machine Learning*, 45, pp. 5–32. doi: https://doi.org/10.1023/A:1010933404324.

Calvert, J. S., Price, D. A., Chettipally, U. K., Barton, C. W., Feldman, M. D., Hoffman, J. L., Jay, M. and Das, R. (2016) 'A computational approach to early sepsis detection', *Computers in Biology and Medicine*, 74, pp. 69–73. doi: 10.1016/j.compbiomed.2016.05.003.

Che, Z., Purushotham, S., Cho, K., Sontag, D. and Liu, Y. (2018) 'Recurrent Neural Networks For Multivariate Time Series With Missing Values', *Scientific Reports*, 8(6085).

Chen, T. and Guestrin, C. (2016) 'XGBoost: A scalable tree boosting system', *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. doi: 10.1145/2939672.2939785.

Chollet, F. (2017) *Deep Learning with Python*. Manning.

Cowie, M. R., Blomster, J. I., Curtis, L. H., Duclaux, S., Ford, I., Fritz, F., Goldman, S., Janmohamed, S., Kreuzer, J., Leenay, M., Michel, A., Ong, S., Pell, J. P., Southworth, M. R., Stough, W. G., Thoenes, M., Zannad, F. and Zalewski, A. (2017) 'Electronic health records to facilitate clinical research', *Clinical Research in Cardiology*, 106(1–9). doi: 10.1007/s00392-016-1025-6.

Desautels, T., Calvert, J., Hoffman, J., Jay, M., Kerem, Y., Shieh, L., Shimabukuro, D., Chettipally, U., Feldman, M. D., Barton, C., Wales, D. J. and Das, R. (2016) 'Prediction of Sepsis in the Intensive Care Unit With Minimal Electronic Health Record Data: A Machine Learning Approach', *JMIR Medical Informatics*. JMIR Publications Inc., 4(3), pp. 1–15. doi: 10.2196/medinform.5909.

Du, J. A., Sadr, N. and De Chazal, P. (2019) 'Automated Prediction of Sepsis Onset Using Gradient Boosted Decision Trees', in *2019 Computing in Cardiology Conference (CinC)*. doi: 10.22489/CinC.2019.423.

Fleuren, L. M., Klausch, T. L. T., Zwager, C. L., Schoonmade, L. J., Guo, T., Roggeveen, L. F., Swart, E. L., Girbes, A. R. J., Thoral, P., Ercole, A., Hoogendoorn, M. and Elbers, P. W. G. (2020) 'Machine learning for the prediction of sepsis: a systematic review and meta-analysis of diagnostic test accuracy', *Intensive Care Medicine*, 46, pp. 383–400. doi: 10.1007/s00134-019-05872-y.

Futoma, J., Hariharan, S., Heller, K., Sendak, M., Brajer, N., Clement, M., Bedoya, A. and O'brien, C. (2017) 'An Improved Multi-Output Gaussian Process RNN with Real-Time Validation for Early Sepsis Detection', in *Machine Learning for Healthcare Conference*.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. MIT Press. Available at: http://www.deeplearningbook.org.

He, H. and Garcia, E. A. (2009) 'Learning from imbalanced data', *IEEE Transactions on Knowledge and Data Engineering*, 21(9), pp. 1263–1284. doi: 10.1109/TKDE.2008.239.

Hochreiter, S. and Schmidhuber, J. (1997) 'Long Short-Term Memory', *Neural Computation*, 9(8), pp. 1735–1780.

Johnson, A. E. W., Aboab, J., Raffa, J. D., Pollard, T. J., Deliberato, R. O., Celi, L. A. and Stone, D. J. (2018) 'A comparative analysis of sepsis identification methods in an electronic database HHS Public Access', *Critical Care Medicine*, 46(4), pp. 494–499. doi: 10.1097/CCM.0000000000002965.

Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L. W. H., Feng, M., Ghassemi, M., Moody,

B., Szolovits, P., Anthony Celi, L. and Mark, R. G. (2016) ‘MIMIC-III, a freely accessible critical care database’, *Scientific Data*. Nature Publishing Groups, 3. doi: 10.1038/sdata.2016.35.

Karpathy, A. (2015) *The Unreasonable Effectiveness of Recurrent Neural Networks*. Available at: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (Accessed: 18 March 2020).

Kreyszig, E. (2011) *Advanced engineering mathematics*. 10th ed. John Wiley Sons Inc.

Levy, M. M., Artigas, A., Phillips, G. S., Rhodes, A., Beale, R., any Osborn, T., Vincent, J.-L., Townsend, S., Lemeshow, S. and Phillip Dellinger, R. (2012) ‘Outcomes of the Surviving Sepsis Campaign in intensive care units in the USA and Europe: a prospective cohort study’, *Infectious Diseases*, 12(12), pp. 919–924. doi: 10.1016/S1473-3099(12)70239-6.

Mao, Q., Jay, M., Hoffman, J. L., Calvert, J., Barton, C., Shimabukuro, D., Shieh, L., Chettipally, U., Fletcher, G., Kerem, Y., Zhou, Y. and Das, R. (2018) ‘Multicentre validation of a sepsis prediction algorithm using only vital sign data in the emergency department, general ward and ICU’, *BMJ Open*. BMJ Publishing Group, 8(1). doi: 10.1136/bmjopen-2017-017833.

MIT Critical Data (2016) *Secondary Analysis of Electronic Health Records*, *Secondary Analysis of Electronic Health Records*. Springer International Publishing. doi: 10.1007/978-3-319-43742-2.

Mitchell, T. M. (1997) *Machine Learning*. McGraw-Hill.

Moor, M., Horn, M., Rieck, B., Roqueiro, D. and Borgwardt, K. (2019) ‘Early Recognition of Sepsis with Gaussian Process Temporal Convolutional Networks and Dynamic Time Warping’, in *Machine Learning Research*.

Morrill, J., Kormilitzin, A., Nevado-Holgado, A., Swaminathan, S., Howison, S. and Lyons, T. (2019) *The Signature-Based Model for Early Detection of Sepsis from Electronic Health Records in the Intensive Care Unit, 2019 Computing in Cardiology Conference (CinC)*. doi: 10.22489/cinc.2019.014.

Olah, C. (2015) *Understanding LSTM Networks -- colah’s blog*. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (Accessed: 18 March 2020).

Paoli, C. J., Reynolds, M. A., Sinha, M., Gitlin, M. and Crouser, E. (2018) 'Epidemiology and costs of sepsis in the United States-an analysis based on timing of diagnosis and severity level', *Critical Care Medicine*. Lippincott Williams and Wilkins, 46(12), pp. 1889–1897. doi: 10.1097/CCM.0000000000003342.

Pollard, T. J. and Celi, L. A. (2017) 'Enabling Machine Learning in Critical Care.', *ICU management & practice*, 17(3), pp. 198–199.

Reyna, M. A., Josef, C. S., Jeter, R., Shashikumar, S. P., Westover, M. B., Nemati, S., Clifford, G. D. and Sharma, A. (2020) 'Early Prediction of Sepsis From Clinical Data: The PhysioNet/Computing in Cardiology Challenge 2019', *Critical care medicine*, 48(2), pp. 210–217. doi: 10.1097/CCM.0000000000004145.

Saito, T. and Rehmsmeier, M. (2015) 'The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets', *PLOS ONE*. Edited by G. Brock, 10(3), p. e0118432. doi: 10.1371/journal.pone.0118432.

Samuel, A. L. (1959) 'Some Studies in Machine Learning Using the Game of Checkers', *IBM Journal of Research and Development*, 3(3), pp. 210–229. doi: 10.1147/rd.33.0210.

Seymour, C. W., Liu, V. X., Iwashyna, T. J., Brunkhorst, F. M., Rea, T. D., Scherag, A., Rubenfeld, G., Kahn, J. M., Shankar-Hari, M., Singer, M., Deutschman, C. S., Escobar, G. J. and Angus, D. C. (2016) 'Assessment of clinical criteria for sepsis for the third international consensus definitions for sepsis and septic shock (sepsis-3)', *JAMA - Journal of the American Medical Association*. American Medical Association, 315(8), pp. 762–774. doi: 10.1001/jama.2016.0288.

Stanford Vision and Learning Lab (2019) *CS231n Convolutional Neural Networks for Visual Recognition*. Available at: http://cs231n.github.io/neural-networks-1/ (Accessed: 18 March 2020).

U.S. Department of Health & Human Services (2020) *Largest Study of Sepsis Cases among Medicare Beneficiaries Finds Significant Burden*. Available at: https://www-hhs-gov.cdn.ampproject.org/c/s/www.hhs.gov/about/news/2020/02/14/largest-study-sepsis-cases-among-medicare-beneficiaries-finds-significant-burden.html?amp (Accessed: 17 February

2020).

Vincent, J. L., Moreno, R., Takala, J., Willatts, S., De Mendonça, A., Bruining, H., Reinhart, C. K., Suter, P. M. and Thijs, L. G. (1996) 'The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure', *Intensive Care Medicine*, 22(7), pp. 707–710. doi: 10.1007/BF01709751.

Weis, S., Dickmann, P., Pletz, M. W., Coldewey, S. M., Gerlach, H. and Bauer, M. (2017) 'The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)', *Deutsches Arzteblatt International*. Deutscher Arzte-Verlag GmbH, 114(29–30), pp. 801–810. doi: 10.1001/jama.2016.0287.

Zabihi, M., Kiranyaz, S. and Gabbouj, M. (2019) 'Sepsis Prediction in Intensive Care Unit Using Ensemble of XGboost Models', in *Computing in Cardiology Conference (CinC)*. doi: 10.22489/CinC.2019.238.