Universiteit Leiden

Master Computer Science

Using AI to predict ICU patient mortality.

Name:	Jelle van den Berg
Student ID:	s2072106
Date:	June 24 2019
Specialisation:	Computer Science and Advanced Data Analytics
1st supervisor:	Dr. Kaifeng Yang
2nd supervisor:	Dr. Wojtek Kowalczyk

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

In today's hospitals, there is a growing amount of advanced tools that gather data. A section of these data are used by doctors to monitor a patient's condition or support in diagnosing a patient, however, the majority of these data are untouched and unused. These data can be used to improve health care using state of the art techniques from Artificial Intelligence and Machine Learning. In this thesis, we've picked the mortality prediction model as a subject to improve. The current model can only be used after the first 24 hours of a patient's hospital stay, is only able to model linear relationships, is equal for all hospitals all over the world and uses only a small part of the available data. We've created a new mortality prediction algorithm that is able to make more accurate predictions than the current standard in mortality prediction, is able to predict on a continuous scale and has an explainable result. We propose a model architecture that is hospital independent and can be used in all hospitals. We've developed an interface where clinicians can consult the model and we propose a method to test the impact of our model.

Contents

1 Introduction

The Intensive Care Unit, further called the ICU, is the hospital unit for the most critically injured or ill patients. Patients are transferred here when their status is critical or their chance to survive becomes endangered. Therefore the ICU has a wide variety of advanced tools to measure the patient's condition. All these measurements are saved and result in huge sets of medical data. We think these data can be used to enhance the patient's health care by improving the mortality prediction model. For example, patients that are predicted so have a high probability of dying could be treated more intensively. The mortality prediction model predicts the chance of patient mortality. This is the chance that a patient will die during his/her hospitalization.
Around 30 years ago in the early 1980s, the first mortality prediction models were developed. Some of the models are; Acute Physiology And Chronic Health Evaluation (APACHE) [19], Simplified Acute Physiology Score (SAPS) and Mortality Prediction Model (MPM) all of them predict mortality. These models, although they are updated, still remain the standard. The most updated and widely used model nowadays is the APACHE IV model, of all mentioned mortality models, APACHE IV has the best performance [1]. This model is used as a benchmark against the new model developed in this project. We think that we can develop a model that has more accurate prediction than APACHE IV. One of the reasons is that APACHE IV is based on linear regression, linear regression formulas are fixed so each hospital uses the same formula. It is not unimaginable that different hospitals have different underlying factors that make it hard to use the same score. For example, an academical hospital will have more complex patients than a general hospital has [2] and health care systems differ among countries [32]. Another part where a new model can improve is non-linear relationships. A Linear Regression algorithm by definition can only model linear relationships. For example, leukocyte count has a U-shaped relationship with in-hospital mortality. leukopenia (a decrease in the number of white blood cells) and leukocytosis (white blood cells above the normal range in the blood) are both associated with greater risk of death than a normal count [19].
To cope with more complex relationships in the data more sophisticated Machine Learning algorithms can be used. These algorithms are fed with data and are self-learning. They can learn linear and non-linear relations based on the fed data. The used machine learning technique is called supervised learning where we used a neural network to train our model on historical data. The trained model is then able to make predictions on new data based on historical data.
To prove that the architecture and solution are hospital independent, we've created a single model architecture and tested it on two different datasets. One of which is the LUMC dataset. This is data from the Leiden University Medical Center in Leiden and contains ICU data from patients between 2004 and 2018. The other dataset is the Medical Information Mart for Intensive Care III or MIMIC-III [3]. This is an open dataset provided by MIT containing de-identified health-related data that of patients that stayed on the ICU of the Beth Israel Deaconess Medical Center between 2001 and 2012. The hospitals are further called hospital 1 and 2 to make them anonymous. The dataset of hospital 1 has 20 thousand patients while the dataset of hospital has almost 40 thousand patients. For each hospital, an identical subset of features is extracted, processed and combined into a dataset. Both of the resulting datasets are used to train a model on a similar architecture. The result of this research is a unified approach for predicting ICU patient mortality.

1.1 Goal and Problem

Hospitals and in particular ICU departments have a growing amount of unused data. We think that these data can be used to improve the accuracy of mortality prediction models such as APACHE IV. We want to achieve this by using a Machine Learning model that is trained on historical data. We've set several goals for our project from which we think they are achievable and contribute to better health care. The first goal we've set was to outperform APACHE IV on the first 24 hours. We will specify outperform later as we use multiple metrics to measure this. The second goal of ours was to make a continuous model. In other words, a model that is able to predict after each hour and not only after 24 hours. As the last goal, we developed our model to be descriptive and explainable. A lot of AI models are black boxes where users don't know how the predictions are made by the model. The model we've developed is capable of describing the features and point(s) in time that were decisive in the final prediction.
With these goals achieved, we think we can help to solve one of the big present-day problems in information technology that is; creating information from the vast sets of data. All of the features and data we use in our model are already there and currently unused. We can't ask doctors to analyze these immense datasets as this would be too time-consuming. The solution is to create a smart model that can bundle the data, analyze the data and support in their decision-making process, benefiting both the patients and doctors.

1.2 Structure

After the introduction, we will give a compact analysis and insight into the datasets, the statistical information of both datasets are displayed and the differences between both sets are described. In chapter 3 the methods are described. The used methods like feature selection, preprocessing, model selection, metrics, and optimization techniques are explained. In chapter 4 the experiments and their results can be found including a small analysis of the results. Chapter 5 describes how we've implemented our AI model and how it can be used in the ICU department. In chapter 6 we'll discuss the results and in chapter 7, the conclusion can be found. We end with chapter 8, where we propose improvements that can be made in future research.

2 Datasets

To test our model, there are two datasets. The usage of different datasets shows how the model architecture and our methods perform on different data. The current mortality prediction model, APACHE IV, and similar research [4] and [9] are used to compare our model. The first dataset is a dataset constructed with data from the LUMC ICU from 2004 to 2018. The second dataset is constructed from the MIMIC-III data [3]. Both of these datasets are constructed by filtering the data sources. The hospitals are further called hospital 1 and hospital 2 to keep them anonymous.
To ensure a fair comparison with other research [4], we've used the same filters on our dataset. This meant that we only used the first ICU stay of a patient, excluded patients under the age of 18, and excluded patients that stayed on the ICU for under 4 hours. Below a list of all filter criteria can be found:

- The patient must be 18 or older
- The ICU stay must be the first one of the patient
- The ICU stay must be longer than 4 hours

Patients under 18 years of age were excluded as we required a population with comparable physiological characteristics. Similar to previous studies, patients with less than 4 hours of data were excluded. The data were used to make in-hospital mortality predictions in a manner similar to other mortality scoring algorithms like APACHE IV. All data was anonymized to ensure the patient's privacy was safeguarded. The mortality rate of both datasets after filtering them for valid patients can be found in Table 1. The mortality rate represents the percentage of patients that died during their hospital stay. This data has already been filtered on the criteria stated above.

Table 1: Mortality of both datasets after applying the patient filters.

Dataset	Mortality rate
Hospital 1	18.2%
Hospital 2	12.9%

In Table 2, all used features are described. When a feature has a dataset as a suffix like hosp 1 or hosp 2 it means that the value is only used in that specific dataset. Below this table, both datasets are described in detail and the statistical properties are explained. For each dataset, we've created a table with all the features. We end the chapter with a brief comparison between both datasets.

Table 2: Features used in the research

Variable	Full Name	Description
Static		
Age	-	Age of the patient. Patients over 89 have an obscured age and are all mapped to 89.
Gender	-	Male or female. A 1 for a male and 0 for a female.
Hospital hours	Hours in the hospital before ICU transfer	Hours in the hospital before admission on the ICU.
Emergency	Emergency surgery	Was there an emergency surgery during the ICU admission.
Origin	Originated from	Where did the patient originate from. We categorized this similar to what APACHE IV does. This means that we have four classes. Other, Other Hospital, Floor and OT/Recovery. Other Hospital means that the patient is transferred to the ICU of this hospital. OT/Recovery means that the patient is transferred from the Recovery unit. Floor means that the patient is transferred from a floor within the hospital. The other cases are all categorized as Other.
Readmission	Readmission	If the patient was readmitted to the hospital.
Died	Died during hospital stay	If the patient died during the hospital stay.
Aids	-	If the patient has Aids.
Metastatic carcinoma	-	If the patient has Metastatic Carcinoma.
Immuno - suppression	-	If the patient has Immunosuppression.
Leukemia / Myeloma	-	If the patient has Leukemia/Myeloma.
Cirrhosis	-	If the patient has Cirrhosis.
CRF/HD	-	If the patient has CRF/HD.
System	-	The system where the admission diagnosis was found. E.g. Metabolic, Cardiovascular, Neurologic.
Diagnosis	-	The exact diagnosis that was determined at the admission.
Physiological		
Temperature	-	Temperature of patient.
HR	Heart rate	Heart rate of the patient.
RR	Respiratory rate	Respiratory rate of the patient.
HT	Hematocrit	Hematocrit percentage of the patient.
MAP (hosp 2)	Mean Arterial Blood Pressure	The Mean Arterial Blood Pressure constructed with the Diastolic Blood Pressure (DBP) and Systolic Blood Pressure (SBP).
DBP (hosp 1)	Diastolic Blood Pressure	The Diastolic Blood Pressure of the patient.
SBP (hosp 1)	Systolic Blood Pressure	The Systolic Blood Pressure.
Mechanical ventilation	-	If the person has had mechanical ventilation.
Urine output	OUput of Urine per hour	The output of urine of the person per hour.
GCS (hosp 1)	Glasgow Coma Score	The Glasgow Coma Score is determined based on tests the clinicians do with the patient.
GCS Verbal (hosp 2)	Glasgow Coma Score Verbal	The Glasgow Coma Score is determined based on tests the clinicians do with the patient.
GCS Motor (hosp 2)	Glasgow Coma Score Motor	The Glasgow Coma Score is determined based on tests the clinicians do with the patient.
GCS Eyes (hosp 2)	Glasgow Coma Score Eyes	The Glasgow Coma Score is determined based on tests the clinicians do with the patient.
Labs		
Arterial PH	PH	The (arterial) PH value of the patient.
Urea	-	The amount of urea measured in the patient's blood.
NA+	NA+ or Sodium	The amount of sodium measured in the patient's blood.
BSL	Blood Sugar Level	The blood sugar level measured in the patient's blood
WBC	White Blood cell Count	The white blood cell count measured in the patient's blood.
Creatinine	-	The amount of creatinine measured in the patient's blood.
Bilirubin	-	The amount of bilirubin measured in the patient's blood.
Albumin	-	The amount of albumin measured in the patient's blood.
FIO2	Fraction of Inspired Oxygen	The fraction of inspired oxygen measured in the patient's blood.
PCO2	Partial Pressure of Carbon Dioxide	The Partial Pressure of Carbon Dioxide measured in the patient's blood.
PO2	Partial Pressure of Oxygen	The Partial Pressure of Oxygen. measured in the patient's blood.

2.1 Hospital 1

In Table 3, the statistical information about the unprocessed dataset of hospital 1 can be found. This dataset is already filtered on valid patients and only the first 24 hours are used. The data has not been generalized so the outliers and noise are still visible.

Table 3: Statistics of the unprocessed dataset from hospital 1

Variable name	Unit	Type	Number of measures	Distinct	Mean	Minimum	Maximum	Standard deviation
Static								
Age	Years	Z	471000	75	62	25,5	99,0	14,88
Gender	-	B	471000	2	0,63	0,0	1,0	0,483
Origin	String	Z	471000	4	0,99	0,0	2,50	0,85
Emergency	-	B	471000	2	0,41	0,0	1,0	0,492
Hospital hours	Hours	Z	470016	66	37,40	1,0	101,0	283,78
Readmission	-	B	471000	1	0,01	1,0	0,0	0.08
Died during hospital stay	-	B	471000	1	0,18	1,0	0,0	0.39
Aids	-	B	64610	2	0,14	0	1,0	0,34
Metastatic carcinoma	-	B	67459	2	0,14	0	1,0	0,35
Immunosuppression	-	B	68821	2	0,15	0	1,0	0.35
Leukemia/Myeloma	-	B	67750	2	0,14	0	1,0	0,35
Cirrhosis	-	B	67366	2	0,14	0	1,0	0,35
CRF/HD	-	B	72895	2	0.16	0	1,0	0,36
System	Categorical	Z	471000	6	11,67	0,0	5,0	15,95
Diagnosis	Categorical	Z	471000	216	6,23	0,0	214,5	65,26
Physiological								
Temperature	Celsius	R	48289	45606	35,97	32,0	4,08	1,07
HR	/min	R	391106	41279	82,51	25,0	190,0	1,94
RR	/min	R	337519	21466	17,61	0,0	50,0	5,06
HT	%/100	R	80933	1396	0,34	0,05	23,67	10,65
Arterial PH	PH	R	131512	463	7,35	7,15	7,57	0,08
Mechanical ventilation	-	B	471000	2	0,68	0,0	1,0	0,47
Urine output	ml	R	9085	240	14,43	0,0	425,0	11,91
GCS	Score	R	471000	15	13.597	0	15	3.55
Labs								
Urea	mEq/dL	R	43002	212	8,06	0,0	17,35	4,20
NA+	mEq/dl	R	132326	1328	13,79	100,0	160,0	4,41
BSL	mg/cl	R	135315	1707	15,64	18,02	630,63	57,58
WBC		R	59824	2660	1,26	0,0	2,56	53,56
Creatinine	mg/dl	R	43471	219	11,20	0,0	22,45	0,53
Bilirubin	mg/mL	R	5095	92	16,71	0,0	4,90	1,55
Albumin	g/dl	R	18491	48	2,99	14,0	46,0	6,52
FIO2	%	R	193721	13440	4,83	0,0	9,45	17,34
DBP	mmHg	R	340945	170553	6,02	3,11	8,86	10,99
SBP	mmHg	R	342164	271289	11,57	572	17,31	22,02
PCO2	mmHg	R	130922	1798	39,70	21,43	575,20	7,32
PO2	mmHg	R	87292	3360	10,92	0,0	2,57	60,34

2.2 Hospital 2

In Table 4 the statistical information about the unprocessed dataset of hospital 2 can be found. This dataset is filtered on valid patients and only the first 24 hours are used. The data has not been generalized so the outliers and noise are still visible.

Table 4: Statistics of the unprocessed dataset from hospital 2

Variable name	Unit	Type	Number of measures	Distinct	Mean	Minimum	Maximum	Standard deviation
Static								
Age	Years	Z	921336	73	63,92	18	90	55
Gender	-	B	921336	2	0,57	0	1	0,5
Hospital hours	Hours	Z	921336	84	1,23	0	154	4,46
Emergency	-	B	921336	2	0,11	0	1	0,39
Origin	String	Z	921336	-	-	-	-	-
Readmission	-	B	921336	2	0,82	0	1	0,31
Died during hospital stay	-	B	921336	2	0,13	0	1	0,33
Aids	-	B	921336	2	0,01	0	1	0,09
Metastatic carcinoma	-	B	921336	2	0,03	0	1	0,16
Immunosuppression	-	B	921336	2	0,01	0	1	0,1
Hepatic failure	-	B	921336	2	0,01	0	1	0,11
Leukemia/Myeloma	-	B	921336	2	0,004	0	1	0,06
Cirrhosis	-	B	921336	2	0,04	0	1	0,21
Lymphoma	-	B	921336	2	0,002	0	1	0,04
CRF/HD	-	B	921336	2	0,002	0	1	0,04
Physiological								
Temperature	Celsius	R	93929	2350	36,8	0	37	4,08
HR	/min	R	793266	3948	85	0	366	18
RR	/min	R	427284	1046	11,2	0	127	5,6
HT	%	R	185637	2841	51	-10	46626	198
Arterial PH	PH	R	39104	232	7	6,72	7,69	0,01
GCS eye	Score	R	921336	21	3,25	0	4	1,1
GCS motor	Score	R	921336	30	4,43	0	6	1,65
GCS verbal	Score	R	921336	25	5,43	0	5	1,87
Labs								
Urea	mEq/L	R	65249	237	25,6	0	266	21,5
NA+	mEq/L	R	36002	129	139	0	180	5,9
BSL	mg/dl	R	65012	1017	135,8	0	934	57
WBC	x1000/mm3	R	62508	879	12,3	0	479	10
Creatinine	mg/dl	R	65487	338	1,47	0	255	3,4
Bilirubin	mg/dL	R	17476	396	2,4	0	75	5,14
Albumin	g/dl	R	12339	49	3,04	0	5	0,68
Mechanical ventilation	-	B	95644	2	0.10	1	0	0.31
Urine output	ml	R	293102	817	115,4	0	7300	145,9
FIO2	%	R	120974	292	23,8	0	1001	32,1
MAP	mmHg	R	341353	2828	78,9	-135	9381	32,8
PCO2	mmHg	R	98616	303	41,3	0	151	9,6
PO2	mmHg	R	98619	1291	168,8	0	1914	105

2.3 Comparison of both datasets

If we compare the two datasets, there are a few things that stand out. Perhaps the biggest difference is the size of both datasets. The hospital 1 dataset is almost half the size of the hospital 2 dataset. In Machine Learning it is important to have a big dataset as this will improve the synthesized model. For us, this means that the model would have a lower accuracy on the hospital 1 data than on the hospital 2 data as there is less data to learn from.
In Table 1, there is quite a difference in ICU mortality. In our case this means that the dataset from hospital 1 is more unbalanced than the dataset from hospital 2. We will have to take this in account when designing and evaluating our model. For example, if our model predicts that every patient will survive, it will achieve an accuracy of approximately 81.8% on hospital 1 and an accuracy of 87.1% on hospital 2.
Another difference is in the different features and their units. For example, hospital 1 has Diabolic Blood Pressure (DBP) and Systolic Blood Pressure (SBP) while hospital 2 has the Mean Arterial Pressure (MAP). Another difference is the Glasgow Coma Score (GCS). In hospital 2 there are three separate variables namely the GCS eye score, the GCS motor score, and the GCS verbal score. The sum of these scores is the (total) GCS score. In hospital 1 only the total GCS score is used. Then there are several features that have different units. For example, WBC has a mean of 12 in hospital 2 while in the hospital 1 dataset this is 1,2. This is due to a different unit, this is also the case with BSL where in hospital 2 the mean is 136,00 and in hospital 1 this is 15,64. Other features with a contrasting unit are HT, NA, Urea, Bilirubin, Urine output, and PO2. The difference in units does not matter for the model. It will train on the different datasets and will adapt to the units in that specific dataset. Additionally, the data is normalized so this should not matter for the model.

3 Methods

3.1 Feature selection

The features used in both datasets are chosen in consultation with a clinician from the ICU of the LUMC. In general, it is an extended set of the APACHE IV features. The features contain both static and dynamic features. Static features have a single value during the whole ICU stay, for example, age, gender and admission diagnosis. Dynamic features, for example, heart rate, PH and lab results, have a value that changes from time to time.

We chose to use a combination of general measurements, lab results, patient demographics and admission information as our base set as these are roughly equivalent to what APACHE IV is using. This makes it is easier to compare APACHE IV with our model and it is relatively less noisy than other medical data.

The selected features can be found in Table 2. As mentioned in the previous chapter, we've selected our features based on their presence in the APACHE IV model to make a fair comparison. The base dataset we've used consists of the same features as APACHE IV uses. In Appendix 3, the mapping between the parameter IDs of both data sources can be found. As can be seen, both static and dynamic data are used in the model. This is similar to APACHE IV but it is also our belief that static information like age and gender do have an impact. For example, when a 60-year-old male is transferred to an ICU with a high heart rate, this is different from a 20-year-old female patient with the same heart rate. Features like age, 'died during hospital stay' and 'length of stay prior to ICU stay' were taken from generic tables. The measurements are queried from the general measurement tables with the described IDs.

Other research has proved that high performance (an AUC ROC of 0.934) can be achieved by using physiological data, medicine, and even interventions [9]. We therefore, experimented with additional features to see how they influence the model's performance. The features we used to experiment with are categorized medicines. For hospital 1 we've used the medicine category that is used in the hospital. The medicines are distinguished by their type (e.g. antibiotics, laxatives, anticoagulants, etc.). They are categorized into 28 categories. For hospital 2, we've used the ICD9 category [37], this is the database for the International Classification of Diseases. ICD9 is divided into 18 categories and has subcategories beneath these 18 categories. The medicine we used, are categorized into 88 different subcategories. The medicines are used as one hot encoded features over time. This means between the start time and end time of the medicine prescription, the patient had a 1 as value for the medicine and outside the prescription dates a 0. In our experiments, the model is first tested without the new features and later with the new features included.

3.2 Preprocessing

The first thing we did with both extracted datasets was to split them in a train and test set with a ratio of 0.8 for the training set and 0.2 for the test set. Both train and test sets were still very noisy, in this case meaning that there were outliers in the data and there was missing data. To cope with the noisy data, we preprocessed the train sets in consultation with an expert clinician. We used the so-called $1.5 \times IQR$ technique [5] in combination with the known highest or lowest clinically plausible value to find and clip the outliers. For features as heart rate and temperature, we used a clinically plausible minimum and maximum value. For other values, the IQR or Interquartile range was used. This is the range between the first quartile and the third quartile (see 1). In the $1.5 \times IQR$ technique [5], the IQR is used to determine the minimum and maximum outliers with the equation found in (1), (2) and with the definition of an outlier (3). Q_1 and Q_3 respectively represent the first and third quartile. The first quartile is is the median of the lower half of the data set meaning that 25% of the number lie below Q_1. Q_3 on the other hand is the median of the upper half of the dataset meaning that 75% of the number lie below Q_3.

$$IQR = (Q_3 - Q_1) \tag{1}$$

$$C = 1.5 \times IQR \tag{2}$$

$$outlier = \begin{cases} v < (Q_1 - C) \\ v > (Q_3 + C) \end{cases} \tag{3}$$

When v, a measurement of a certain patient, is bigger or smaller than a certain boundary, the outliers are replaced. When the known data in the hour before the outlier occurred was available (not null) and was not an outlier itself, forward filling is used to replace the outlier. When this was not the case, the value was set to either $Q_3 + C$ or $Q_1 - C$. As Neural Networks cannot operate on missing data like NaNs, the missing values were masked by a -1 as this value is configured as the 'Mask' value in the neural network. More details and information about masking can be found in chapter 3.3. As the processed data with the replaced outliers had different ranges, the data was normalized to improve the performance on Machine Learning tasks [6]. The MinMax scaling, which can be found in equation (4), is used as a normalization technique. This results in scaled features that have values between 0 and 1. The MinMax scaler is fitted on the train data and used to transform both the train and the test set. In the appendix the preprocessed data is visualized using violin plots for respectively the hospital 1 and hospital 2 dataset (see 9 and 9). Please note that the plots represent the preprocessed data.

To use the data in a Long Short-Term Model network, we had to transform it into a 3D matrix. The original dataset was a matrix containing rows for each patient-hour combination. The data was transformed in a matrix with the dimensions of $P \times T \times N$ where P is the number of patients, T is the number of time steps and N is the number of features.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{4}$$

3.3 Model selection

If we summarize our problem we could say that we have two large datasets and a binary classification problem. The large datasets enable us to use neural networks. To achieve a better performance than the current model, we need sufficient training and test data and a suitable model for our problem. Another part of our problem is that we want to use sequences of data and not only a single data point. To predict the mortality of a patient, we don't want to look at a set of data points *in time*. Instead, a more accurate reflection of the patient is given by how the relationship between data points changes *over time*. Therefore we use a type of Recurrent Neural Network (RNN). This is a type of Neural Network that is able to process sequences of data due to their internal architecture. A RNN has a memory that enables it to process sequences of data. This memory is constructed by the nodes inside the network that form a directed graph along a temporal sequence. The RNN should also enable us to make prediction at intermediate timesteps. Figure 1 shows the architecture of a RNN. As can be seen in Figure 1, the first node receives x_0 and outputs h_1. The second node receives x_1 and A_{t-1}, which is the cell state of the last timestep, and output h_1. By combining the new input with the memory, the RNN is able to spot trends in temporal data.

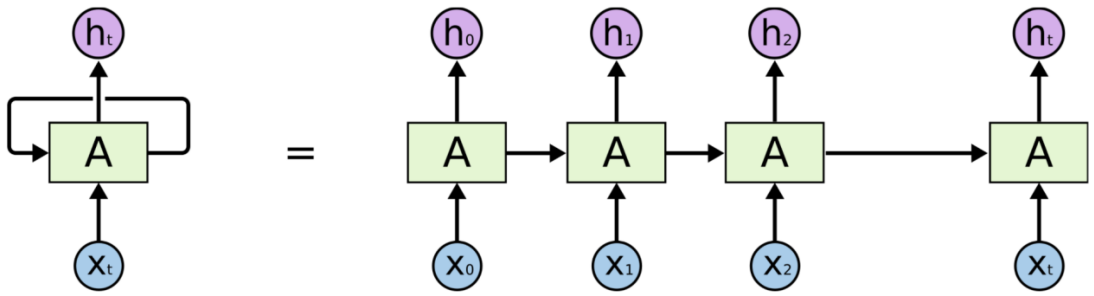


Figure 1: A recurrent neural network. x_t: Input at time t. h_t: Output at time t. Adapted from [11].

The downside of using RNNs is that their memory tends to fade over time. This is caused by the vanishing or exploding gradients problem [7]. In practice, this means that the information from a distant past can no longer be acted upon. Long Short term Memory or LSTM networks, a type of RNN, cope with this problem by using a cell. A common LSTM unit is composed of a cell, an input gate, a forget gate and an output gate. The three gates regulate the data stream in the network. The input gate controls the extent of information flow into the cell. The output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The forget gate controls to which extent the information stays in the cell. The cell is responsible for memorizing the dependencies between the chunks of information in the input sequence [8]. This cell can retain the information for a longer period than a regular RNN can. In Figure 2 the architecture of a LSTM unit is visualized.

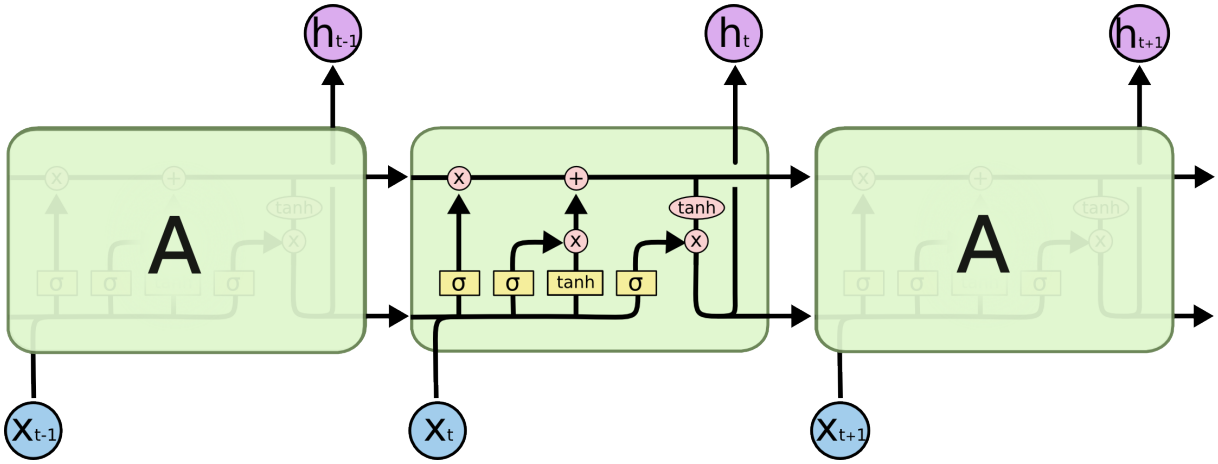
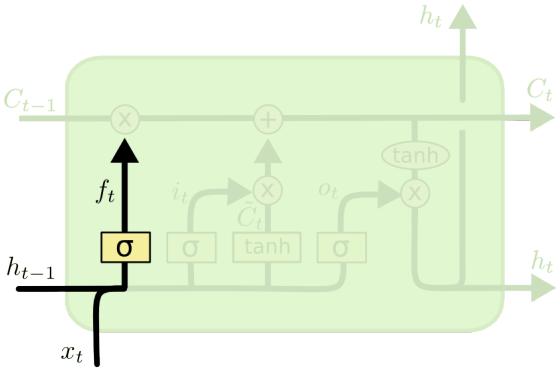


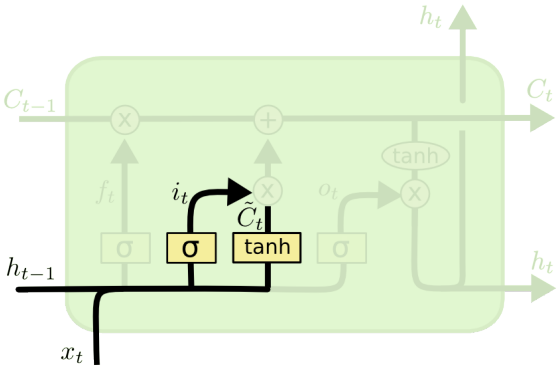
Figure 2: Long Short Term Memory units. Adapted from [11].



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 3: Example of LSTM forget gate. Adapted from [11].

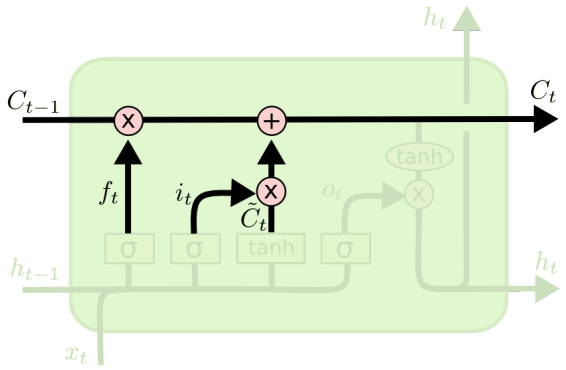
In Figure 3, the forget gate of a LSTM is shown. This gate decides what data should be ignored and forgotten. The output from the previous state, h_{t-1}, together with the input of x_t is led through a sigmoid layer. This layer returns a value between 0 and 1 where a 0 means forget everything and a 1 means remember everything. The result of this is called f_t and is used to calculate the new cell state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 4: Long Short Term Memory input gate. Adapted from [11].

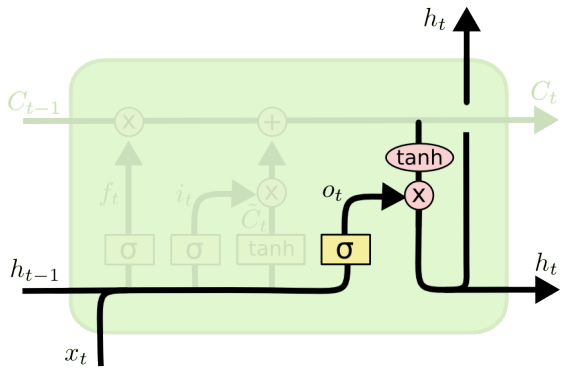
Figure 4 shows the input gate of a LSTM. After the forget layer, x_t and h_{t-1} are led through the input gate (Figure 4). The sigmoid that calculates i_t, decides the values to add to the internal state. The tanh, creates a vector of the new candidate cell state $\tilde{C}_t$ that is later used to calculate the new cell state.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 5: Example of LSTM cell state. Adapted from [11].

In Figure 5, the cell state and its update rule are shown. The new cell state is calculated using the output of the forget gate f_t, the latest cell state C_{t-1} and the outputs of the input gate i_t and $\tilde{C}_t$. In the previous steps, all decisions are made so what rests is two multiplications and addition. The new cell state C_t is then used to calculate the output and is passed on to the next layer.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 6: Example of LSTM output gate. Adapted from [11].

Figure 6 shows the output gate of a LSTM node. In this gate the output of the LSTM cell is calculated. First o_t is calculated with the new input x_t and the previous output h_{t-1} using a sigmoid function. The new cell state C_t is put through a tanh function to transform the values in a range of -1 and 1. The output 0_t is multiplied by result of this tanh function.
For more information about the gates and their functionality please read [11] or [8]. Not only can LSTMs use time series data and cope with vanishing gradients, but they have also been used in a wide range of clinical contexts [4] and have shown to be competitive for mortality predictions [9]. Also, it is better protected against noisy data than other types of RNNs [10]. Therefore we chose to use an LSTM as a model for our problem. As we will build our model on a fixed length of time, we can use a Bidirectional LSTM (B-LSTM). This is a LSTM that not only remembers past states but also future states. A Bidirectional-LSTM achieves this by duplicating the LSTM layer. The original layer gets the input as is and the duplicated layer gets the reversed input. In Figure 7, the architecture of a bidirectional RNN is visualized.

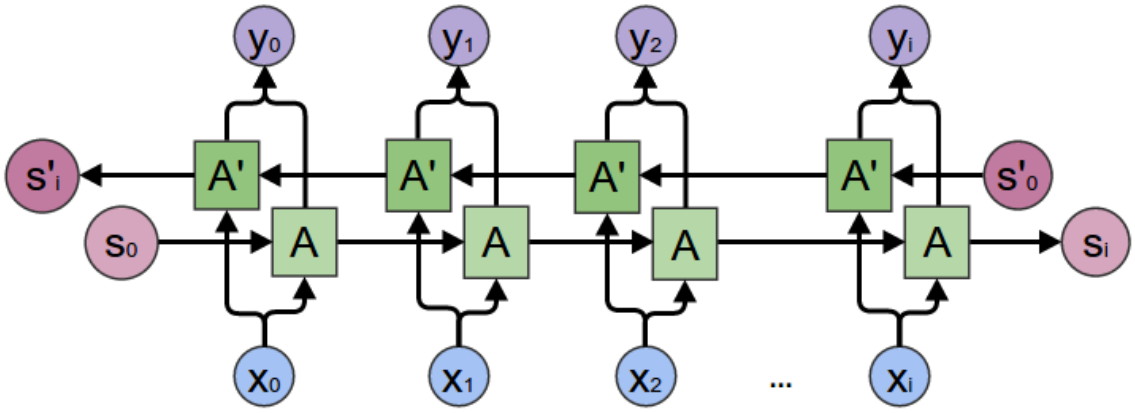


Figure 7: Bidirectional RNN. A': RNN in reverse chronological order, A: RNN in chronological order, x_t: Input at time t. y_t: Output at time t. Adapted from [11].

The advantage of such an architecture is that it can discover trends in a reversed order. When a clinician helps a patient it will look at (historical) trends in the patient's health to assess the risk and make a prediction. However, humans can also think in hindsight. We can learn from the past by understanding and learning about relations. A B-LSTM can have similar properties and can spot trends in reversed order by analyzing future timesteps.
Neural Networks and therefore (B-)LSTMs as well, can be really good at classification tasks but lack explainability. The result is that for most people they are a black box where you put something in and get something out. In a clinical setting, this is not desirable as there are human lives at stake. Therefore an attention layer and SHAP values are used. An attention layer implements an attention model. The attention model is inspired by animals and in particular mammals. Animals have a lot of information, but only focus their attention on a very small part of that information [14]. For example, we have to direct our attention in a landscape full of (social) media and triggers. These models add a layer of neurons in order to visualize the results of the model. The collective output from this layer is used to add a probability to each neuron in this layer, thereby allowing the model to select a combination of neurons that have a higher impact on the final prediction [15]. To implement such a layer, the repository of LzFelix is used [16]. Another technique to improve the explainability of our model is the use of SHAP values [31]. SHAP is an abbreviation for SHapley Additive Explanations. This is a unified approach to explain the output of any machine learning model. SHAP assigns each feature an importance value that represents the effect on the model's prediction of including that feature for a particular prediction. To compute this effect, two models are trained. One model is trained with that feature present, and another model is trained without the feature. Then, predictions from the two models are compared on the current input [31]. This adds another layer of insight into the model. The attention layer provides information on the most important time whereas SHAP provides an insight into the output of the model.

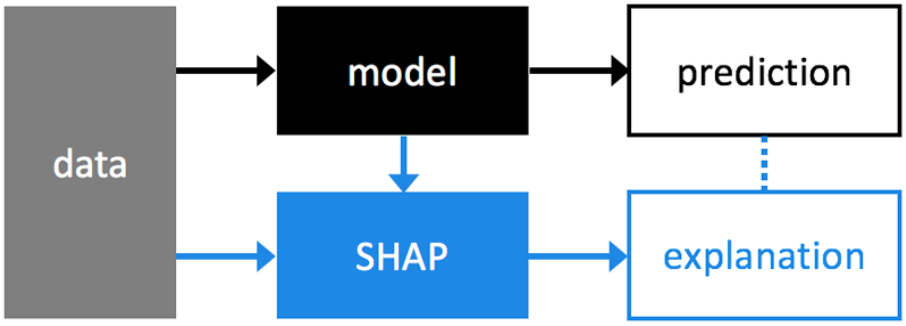


Figure 8: Example of an architecture with SHAP. Adapted from [31].

As a separate experiment(see chapter 3.7.2), a Convolutional Neural Network (CNN) is used to see how such an architecture will perform on time series data. Similar research proved that CNNs are capable of performing

well on time series data [22]. To see if a similar architecture could help us as well, we've tested a CNN model on time series data. Convolutional Neural Networks are a type of deep neural networks that are typically applied to image classification. The architecture of a CNN is similar to the connectivity of neurons in our brain. Specifically the visual cortex. In our brain, the neurons only respond to small pieces of visual stimuli. This is known as the receptive field. There are other neurons who look at the collection of these fields. This structure can be found in the architecture of a CNN. An example can be found in Figure 9. In Figure 9 it is clear that there is a stack of layers. The input image is first led through the first layer and all the way to the fully connected layer at the end. In general, the deeper a CNN is, the more complicated patterns it can recognize. We think that a CNN is capable of finding interesting patterns and believe it should be able to perform on our data.

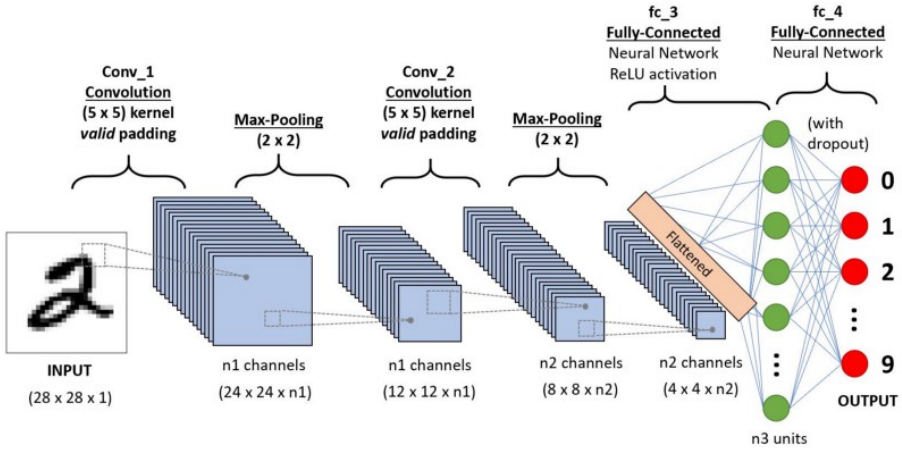


Figure 9: Example of a CNN architecture. Adapted from [25].

The hidden layers have several responsibilities. In general, they compress the input by max-pooling and convolutions of the image [25]. In the network architecture of Figure 9, there are 5 different layers used. The first layer is the input layer which is similar to the input layer of an LSTM. The second layer is a Convolutional layer. These layers are the first filters that are applied to the image. In Figure 10 we can see how a convolutional layer operates. In this example, the input is of a $5 \times 5 \times 1$ dimension. The yellow block is called the filter or kernel and shifts over the image. The kernel moves from left to right with a step/stride of 1. This results in an image of 3×3 called the convolved feature. The convolved feature is passed to the next layer, in the case of Figure 9 this is the max-pooling or pooling layer.

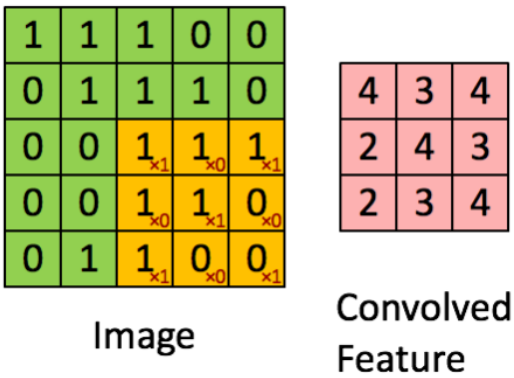


Figure 10: Example of a Convolutional layer. Adapted from [25].

The pooling layer has a similar function as the convolutional layer, namely dimensionality reduction. This will reduce the computational power required to process the data [25]. There are two types of pooling layers: max-pooling layers and average pooling layers. In Figure 11 the difference between these layers is made clear. The max pooling layer takes the maximum value of the pooled area and the average pooling layer takes the average value of the pooled area.

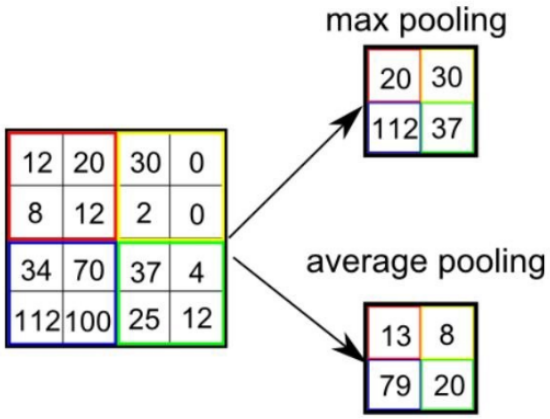


Figure 11: Example of a Pooling layer. Adapted from [25].

The pooling layer moves in a similar fashion over the image as the convolutional filter and takes the maximum out of each kernel position whereas the convolutional kernel takes the sum. Combining these layers with a fully connected layer for output creates a CNN. These architectures can reduce the complexity to process images and are still able to recognize complicated patterns.
Other research has pointed out that CNNs can be reversed to see how a network predicts and what features are important [26]. If a CNN performs well on our problem, this could improve the explainability of our model. According to the research, it could even give new insights into why the model predicts what it predicts.

3.4 Model optimization

The optimization of an AI model is a tedious process that takes a lot of time. To optimize all hyperparameters manually and get the optimal architectures, would take a very long time. We therefore used a plugin called Hyperopt [17], this is a plugin that can automatically optimize the hyperparameters. To optimize our model and architecture, we chose to optimize the number of LSTM units, the number of layers, the amount of dropout and the number of neurons in the added layers for our main model. For the CNN model, we chose to optimize the number of convolutional filters, the size of the convolutional kernels and the size of the max pooling kernels. For the exact configuration of Hyperopt please read chapter 3.7. By looking at other studies, a rough estimation of bounds can be set for the optimal parameters. It is still impossible to know the optimal settings *a priori*. When looking for the exact optimal parameter settings, one has to keep in mind that all parameters can have some sort of interaction with each other. This makes the number of possibilities way too large for a human to try. Therefore Hyperopt is used. Hyperopt is a library for python built for optimization in huge search spaces, we've used the FMin function of this library to optimize our model. It uses a kind of Bayesian optimization. This approach is efficient on problems where evaluating the objective function, our model, is computationally expensive. To evaluate a single composition of parameters, the whole model needs to be retrained and evaluated. Hyperopt constructs a probability model of the objective function, in our case the model's F_1 score on the validation set. This is also called a surrogate. The surrogate is constructed with the Tree Parzen Estimator or TPE algorithm. The usage of a surrogate is relatively cheap to optimize, especially compared to the original objective function. The surrogate is used to experiment with different combinations of parameters. The strategy of selecting the parameter configuration is a case of exploration v.s. exploitation.

Meaning that there should be a healthy balance between exploiting promising results on one side and exploring different results on the other side. Hyperopt first tries different settings on the surrogate and then tries the best setting on the objective function. The result is used to further update the surrogate. As the surrogate becomes more updated, it becomes more accurate in its predictions and the best parameter settings can be tested relatively cheap. In general a Hyperopt optimization consists of three components. The first component is the objective function, in our case this is the $F_!$ score on the validation set. The second component is the domain space, this is the range and configuration of hyperparameters that we pass to Hyperopt. The last component is the optimization algorithm, in our case this is the TPE algorithm.
Overfitting is the process of training a model specifically for a given set. A simple example would be that the model learns patterns that are only present in the training data but not in the test data. The result is that the model performs really good on the train data but not on the test data. In our case, the model would be learning patterns from the historical dataset that are not representative for actual patients. To reduce overfitting, two techniques are used namely: dropout and early stopping. Dropout is a technique to avoid overfitting. It 'drops out' hidden units during the training of a network. The term 'drop out' refers to ignoring units or neurons during training and thus not updating their weights during a forward or backward pass. In network design, the dropout rate is a probability of ignoring these neurons. For example, a dropout, d of 0.3 means that all neurons have a chance of $1-d$ of not dropping out. This process forces a network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons [18]. Early stopping is another technique to avoid overfitting. This technique monitors the generalization error and stops training when it starts going up. This means that whenever the error on the test set is growing and the error on the train set is declining, the model is overfitting on the training data. The process of overfitting is visualized in Figure 12.

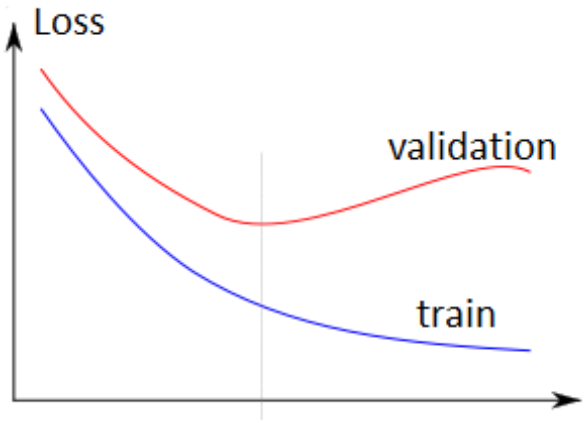


Figure 12: Visualization of overfitting. The red line represents the validation loss and the blue line the training loss. At the start, both of them are declining. Where the grey marker is, the red and blue line starts to diverge. The error on the validation set becomes larger while the error on the training set is still declining. This process is called overfitting. Adapted from [20].

3.5 Model Architecture

We've created two model architectures for our experiments. Our main model, a LSTM architecture, and a CNN architecture. Both architectures are optimized using Hyperopt. For the exact configuration of Hyperopt please read chapter 3.7. The main model consists of five different layers. The composition of these layers can vary depending on the Hyperopt configuration. Starting at the input layer we then have the masking layer, bidirectional LSTM layer, the attention layer and finally the dense output layer. Another composition that we tested, adds a dense layer in between the LSTM and the attention layer. Both architectures can be found in Figures 13 and 14.

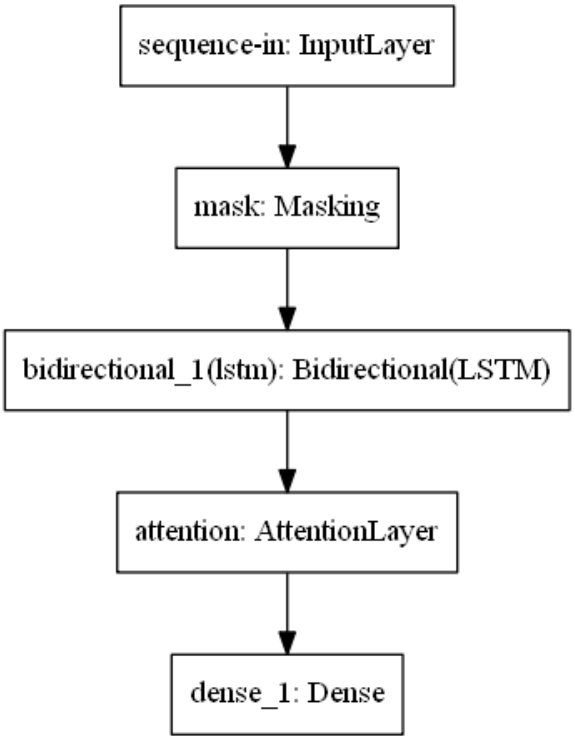


Figure 13: First configuration of layers of the main model.

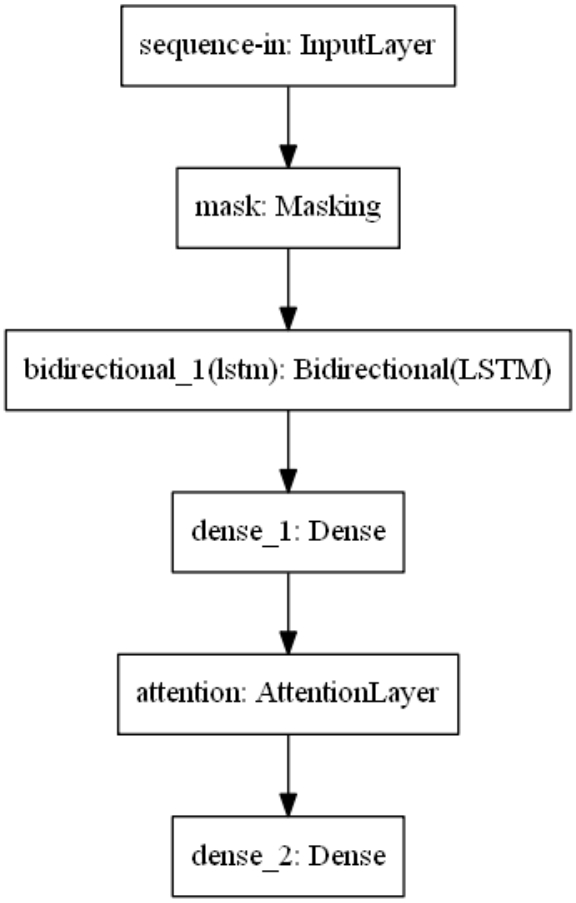


Figure 14: Second configuration of layers of the main model.

The input layer receives the data and maps it to a tensor [12]. This layer expects an input shape with the dimensions $P \times T \times N$ where P is the number of samples or patients, in this case, T is the number of timesteps and N is the number of used features. The data that was masked during the preprocessing should not be used by the network. Therefore we have a masking layer. This layer expects a value that is used as the masking value. When all features in a specific timestep are equal to the mask value, -1 in our case, the time step is skipped. The result is fed to the Bidirectional LSTM layer. This layer expects the number of LSTM units that it should use and the dropout factor that it should use (for more on dropout please read chapter 3.4). In this layer the weight computation takes place. The result is forwarded to the attention layer. Here a layer of neurons is added to specify the most important timesteps in the final decision. Finally, the Dense output layer transforms its input into a probability of mortality. It does so by using softmax in a vector with two values. The first value is representing the chance that the patient will survive and the second layer is representing the chance that the patient will die. The model uses a categorical cross entropy loss function with class weights. The class weights penalize the false positives. For more information on the classification classes please read chapter 3.6.

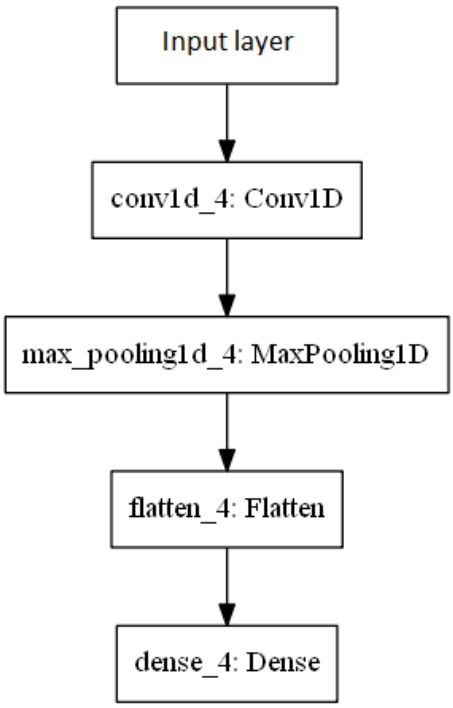


Figure 15: Layer architecture of the CNN model.

In Figure 15 the layered architecture of the used CNN can be found. The input and output layers are similar to the LSTM network. This makes sense as the network has the same input and output. As input a processed matrix of patient data. As output a vector containing the chance that the patient will survive and the chance that the patient will die. In the second layer, the input passes through a convolutional layer. This layer has several filters with kernels of fixed size. The exact amount of filters and sizes are selected by Hyperopt during our experiments. The convolved features are passed to the max pooling layer. Here the results are filtered on the maximum values in the images. The results are then flattened and given to the output layer. The model uses a categorical cross entropy loss function with class weights. The class weights penalize the false positives. For more information on the classification classes please read chapter 3.6.
One big difference between the two models is the absence of a masking layer in the CNN. As of yet, there is no technique to support value masking in CNNs. The result is that the model does not skip unavailable hours (hours without a single value) as the LSTM does. It is also worth mentioning that the model is relatively shallow for a CNN. Most CNNs have multiple convolutional and max pooling layers stacked while we only use a single layer for each of them.
Next to the hyperparameters that we optimize using Hyperopt, there are also some parameters that we've set manually. For example the model is trained using the RMSPROP optimizer with default configuration(learning rate=0.001, rho=0.9). The weight initialization of all layers are left at the default setting. This means that all layers initialize their weights using the glorot uniform distribution. Other parameters are all left at the default configuration. These configurations can be found in the Keras [12] documentation.

3.6 Model Evaluation

To evaluate the model several metrics are used. As in this problem, the data is unbalanced so a standard metric like accuracy would not suffice. The model can have relatively high accuracy when it predicts that every patient will survive as the data is imbalanced. This means that accuracy alone isn't a valid indicator of the model's performance. Especially in this problem, making a wrong prediction is very dangerous. For example, when our model predicts that the patient will survive, it is possible that the doctor allocates his attention different than when the model predicts a patient will die. It is therefore crucial that our model balances its predictions. To avoid these cases we selected multiple metrics that all serve and show different performances of our model. In total, we used six different metrics. Other research in this domain used the AUC ROC as most important metric but we think that a combination of multiple metrics provides a better overview of the results. The used metrics are accuracy, precision, F_1 score, specificity, sensitivity, Area Under the Curve of Receiver Operating Characteristic (AUC ROC) and confusion matrices. All of these metrics will be explained. For some of the metrics, it is helpful to understand the classification classes. As this is a binary problem, the model predicts either if the patient will survive or not. These predictions can be right or wrong. We can use the classification classes to identify how our model performs and how it is balanced. Below you can find a list of the classification classes for our problem:

- True Positive (TP) = The number of patients classified as surviving that actually survive.
- False Positive (FP) = The number of patients classified as surviving that actually die.
- True Negative (TN) = The number of patients classified as dying that actually die.
- False Negative (FN) = The number of patients classified as dying that actually survive.

3.6.1 Accuracy

Accuracy is one of the most used metrics. When we have a set of true values and a set of predictions, the predictions can be called accurate if the average of the predictions is close to the true value. The accuracy can be perceiving, as a high value doesn't mean that the model is performing well. For example, if we have 1 million patients and 10% of them die in the hospital. When we let our model predict how many patients die and it predicts that all of them survive, our model will achieve an accuracy score of 90%. When we look into the score, we see that indeed in 90% of the cases it is right but the other 10% are false positives. This behavior occurs when using imbalanced sets. In our case, we could get around 85% accuracy on the sets by predicting that every patient will survive. As this is not desirable, we will use other metrics in addition to the accuracy to get a better overview of the real model performance. The accuracy can be calculated with the following equation (5).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

3.6.2 Precision

Precision is a metric that looks at the total positively predicted samples and looks at which fraction is correct. For example, when we have a lot of patients that are predicted to die while only a small fraction will actually die, we will have low precision. Precision is often used in combination with recall. Together they form a measure of relevance for the predicted data. The precision can be calculated with the following equation (6).

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

3.6.3 F_1 score

The F_1 score is a performance measure that uses precision and recall/sensitivity to calculate a score. The score can be described as the harmonic average between precision and recall. A F_1 score of 1 means that the model has perfect precision and recall. The F_1 score can be calculated with the following equation (7).

$$F_1 = \frac{recall^{-1} + precision^{-1}}{2} = 2 * \frac{recall * precision}{recall + precision} \tag{7}$$

The F_1 score got criticized as it gave equal importance to recall and precision [34]. This is not desirable in all cases. For example, in medical problems, it can be that recall has higher importance than precision has. Therefore the F_β score was invented. The F_β score introduces a parameter β to control the importance of recall and precision. The β is a value between 0 and 1. Instead of using $\frac{1}{2}$ precision and $\frac{1}{2}$ recall, it lets you specify by $\frac{\beta}{\beta+1}$ for recall and $\frac{1}{\beta+1}$ for precision. The F_β score can be calculated with the following equation.

$$F_\beta = (1 * \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall} \tag{8}$$

3.6.4 Specificity

The Specificity is also called the True Negative Rate (TNR). It contains the proportion of negatively classified samples that are actual negatives. In our case, this is the percentage of people classified as being healthy against all people that are actually healthy. In other words, it is the rate at which the model can tell us that a person is certain to survive. Having a high specificity, close to 1, means that there are few false alarms. Having a low Specificity means that there are a lot of false alarms. Below the equation to calculate the specificity is visualized.

$$Specificity = \frac{FN}{FN + FP} \tag{9}$$

3.6.5 Sensitivity

The Sensitivity is also called the True Positive Rate (TPR). It contains the proportion of positively classified samples that are actually positive. For example, when we predict people dying. It is the percentage of people predicted to die that is correct. In the medical world, it is important to have a high sensitivity as this means that no patients that actually die are predicted to survive. Therefore in the medical world, this metric is sometimes called the detection rate. In almost all cases, there is a balance between specificity and sensitivity. In our case, we prefer to have a higher sensitivity to avoid people slipping under the radar. To achieve this, class weights are used to penalize False positives. The class weights are calculated automatically using the sklearn package [13]. The sensitivity can be calculated with the following equation:

$$Sensitivity = \frac{TP}{TP + FN} \tag{10}$$

3.6.6 AUC ROC

The Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classification problem. The ROC curve is created by plotting the TPR or Sensitivity against the False Positive Rate (FPR). The FPR can be calculated using the Specificity as 1 - Specificity. On the y-axis, the Sensitivity can be found while on the x-axis the FPR is visualized. The plotted curve starts in the bottom left corner where sensitivity is 0 and specificity is 1 (as x-axis represents 1 - specificity) and ends in the upper right corner where the sensitivity is 1 and the specificity is 0. When the curve is a straight diagonal, the model has no capacity to separate classes and will achieve the same result as a random separation. The more the curve bends to the upper left corner, the better the balance between both metrics are and the better the model scores are. In Figure 16 an example of a ROC curve can be found.

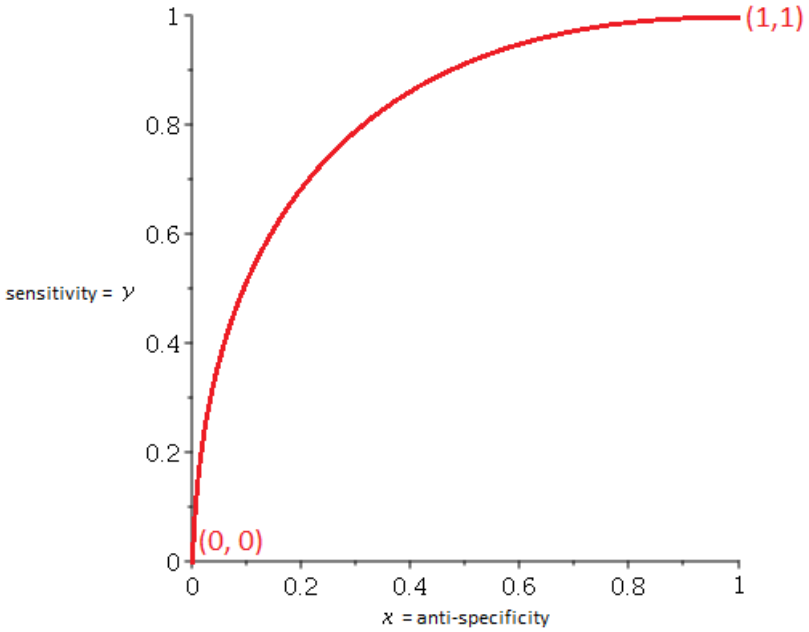


Figure 16: Example of an ROC curve. Adapted from [21].

The AUC ROC is the Area Under the ROC curve. This metric shows the degree of separability by the model. A value close to 1 means that the model is able to separate both classes perfectly while a lower value means that it has a higher rate of errors. This measure can be used as a metric as it is not sensitive to class imbalance as opposed to accuracy for example.

3.6.7 Confusion Matrix

The Confusion Matrix is a visual metric that can help to interpret model results. The confusion matrix shows how much true positives, true negatives, false positives, and false negatives the model predicts. In our case where there is a higher cost/penalty at predicting a false positives than to predict a false negative, this visualizes how the predictions are located. This metric gives a visual representation of the results.

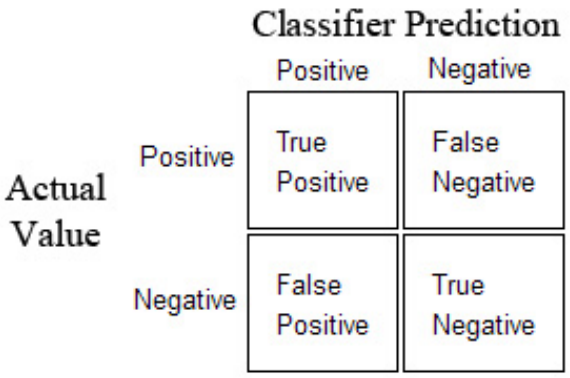


Figure 17: Example of a Confusion matrix. Adapted from [27].

Figure 17 shows a Confusion Matrix that visualizes the distribution of the classification classes in the matrix.

3.7 Experiments

All experiments are executed with a Hyperopt implementation. The Hyperopt script was provided with a domain or search space that is described in chapter 3.7 and had 10 evaluations to find an optimal model. The Hyperopt script is set to optimize the F_1 score on the validation set, the model with the best score was returned by the script. This model was then used to capture the performance of the model. This is done out of precaution for randomness.
To find the optimal model and number of hours to train our model on, we did several experiments. Our experiments all have a matching structure. We first created a Hyperopt script that we provided with the task to find the best F_1 score on the validation set. The domain space that was provided to the script are the hyperparameter configurations that can be found below. We configured Hyperopt to run 10 evaluations. Afterwards, it returned the model with the best performance. The model returned by the Hyperopt script is used to capture the performance. This structure is used for all of our experiments. The idea is that by first creating an optimal model based on 10 evaluations and then evaluating it, we will exclude randomness and can give a reliable result. The first experiment we did was with 24 hours of data. We focused on the descriptiveness of the model by creating plots of the attention layer, the features and the SHAP values. The second experiment we did was with 100 hours of data. We tested it on the datasets of both hospitals with and without medicine. We also tested the CNN on 100 hours of data. The last experiment we did was on 168 hours of data. Each experiment has a different number of hours to work with. The idea is that this gives us insight in the number of hours that we should use for our final model. In Tables 5 and 6 the parameters of Hyperopt are visualized. The configuration ranges from Table 5 are used in experiment 1, 2 and 3. The configuration ranges from Table 6 are used in the CNN experiment from experiment 2.
The optimization problem we have can be divided in three parts. First we have the objective function that Hyperopt is optimizing. This is the F_1 score on the validation set. The domain space is the configuration of Hyperopt that can be found in Tables 5 and 6. The optimization algorithm that we've selected is Tree Parzen Estimator (TPE).

Table 5: Hyperopt configuration ranges for the LSTM in experiment 1, 2 and 3

Setting	Range
LSTM units	[4, 200] (integers)
Dropout on LSTM layer	[0, 1]
Use extra dense layer (boolean)	true, false
Number of units on dense layer	[4, 24]

Table 6: Hyperopt configuration ranges for CNN in experiment 2

Setting	Range
Number of convolutional filter	[60, 160] (integers)
Kernel size of convolutional filters	[3, 48] (integers)
pool size of maxpooling layer	[2, 24] (integers)

3.7.1 Experiment 1

In the first experiment, we are using our AI model to improve upon the existing APACHE IV scores. In this experiment we mimicked the method of APACHE IV, to do this we used similar features as APACHE does and only used the first 24 hours of patient data. To get the optimal data, a Hyperopt script is generated to construct an optimal model. The scores from our model are then compared to the APACHE IV scores from the database of hospital 1.
Another experiment we did is adding medicine to the dataset to see what this does to the overall performance. As we described in chapter 3.1, other research with promising results also used data like interventions and medicine. We, therefore, experimented with the additions of medicine.

3.7.2 Experiment 2

In the second experiment, we tested our model on more than just a single day. This is an improvement over APACHE IV. The approach for the experiment is similar to the previous one. We've configured Hyperopt to find an optimal model configuration. We chose 100 hours as an experiment as 75% of the patients stays under 100 hours in total on the ICU.
Another experiment is the usage of a Convolutional Neural Network (CNN). As described in the model selection chapter 3.3, research [22] proved that CNNs are capable of achieving promising performances on time series data.

3.7.3 Experiment 3

The biggest time sequence we used in the research is a sequence of 168 hours. This is an exact week after the patient's admission on the ICU. We chose to use 168 hours as 85% of all patients stay under a week on the ICU. Therefore this is a good amount of hours to use in our final model.

4 Results

In this section, the various experiments and their results are described. All experiments are executed with a Hyperopt implementation. The Hyperopt script was provided with a domain or search space that is described in chapter 3.7 and had 10 evaluations to find an optimal model. The Hyperopt script is set to optimize the F_1 score on the validation set, the model with the best score was returned by the script. This model was then used to capture the performance of the model. This is done out of precaution for randomness.

4.1 Experiment 1: 24 hours

The first experiment mimicked the method of APACHE IV. It used similar features and only the first 24 hours of data. After preprocessing the data, a model was designed that could train on the data. To get the optimal data, a Hyperopt script was generated to construct an optimal model. This model was then used on hospital 1 and hospital 2. The Apache scores are gathered from the database of hospital 1 and represent the APACHE IV scores of ICU patients from hospital 1. The optimal model was used to generate the results that can be found in Table 8. In Table 7 the optimal model is described.

Table 7: Optimal model settings as returned by our Hyperopt script

Setting	Range	Hospital 1	Hospital 1 medicines	Hospital 2	Hospital 2 medicines
LSTM units	[4, 200] (integers)	48	140	100	100
Dropout on LSTM layer	[0, 1]	0.2	0.2	0.3	0.3
Use extra dense layer (boolean)	true, false	no	no	no	no
Number of units on dense layer	[4, 24] (integers)	-	-	-	-

Table 8: Results of first 24 hours

Metric	Hospital 1	Hospital 1 medicines	Hospital 2	Hospital 2 medicines	APACHE
Accuracy	0.80	0.79	0.85	0.89	0.82
Precision	0.80	0.79	0.85	0.89	0.49
F1 score	0.71	0.70	0.72	0.78	0.41
Sensitivity	0.77	0.93	0.95	0.96	0.87
Specificity	0.49	0.42	0.44	0.57	0.49
AUC ROC	0.83	0.84	0.89	0.91	0.74

In Table 7 the optimal model configuration is shown. These settings are used in the constructed models from Hyperopt. We have achieved the results from Table 8 using these models. Below in Figure 18 the ROC curves of APACHE and the model on hospital 1 and 2 are visualized in a ROC curve plot. Figures 19, 20 and 21 show the confusion matrices of the experiments.

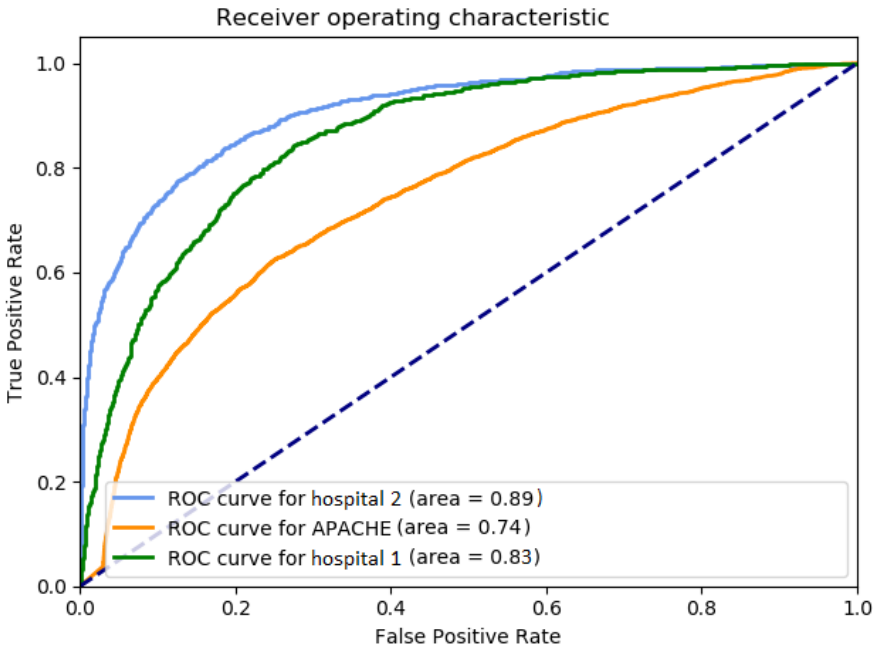


Figure 18: ROC curves of Model on hospital 2 without medicines, Model on hospital 1 without medicines and the APACHE scores from hospital 1. For more information about ROC curves please read chapter 3.6.6.

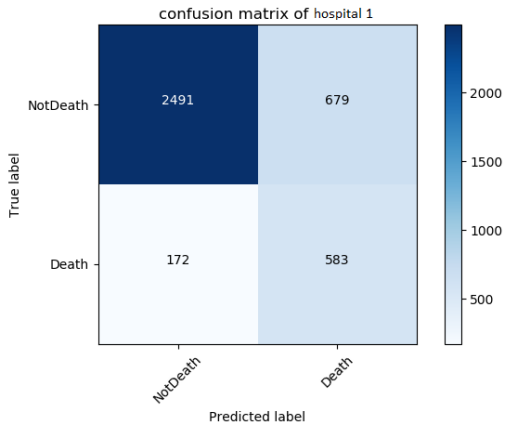


Figure 19: The Confusion matrix of the model on hospital 1

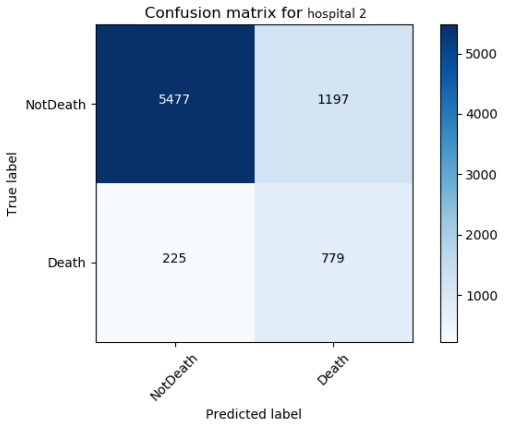


Figure 20: The Confusion matrix of the model on hospital 2

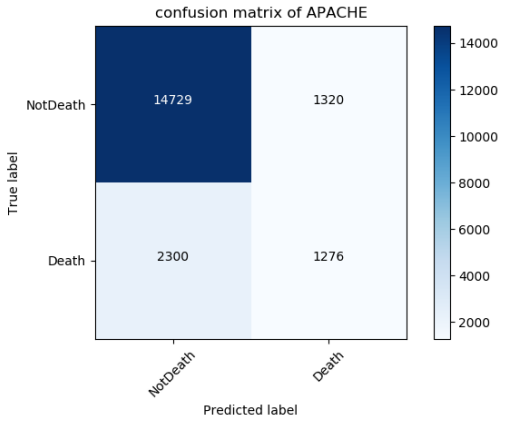


Figure 21: The Confusion matrix of APACHE on hospital 1

As the figures above show, the new models clearly outperform APACHE IV when it comes to ROC and the confusion matrix. The model on hospital 2 achieves an AUC ROC score of 0.89, the model on hospital 1 achieves a score of 0.83 and the APACHE IV predictions only score 0.74. These numbers make clear that our model outperforms APACHE if we look at the AUC ROC score. When we look at the confusion matrices we see the distributions of true positives, false positives, true negatives and false negatives. We can see that they are similar for our model on both the data of hospital 1 and hospital 2. We can see that APACHE has more weight on false positives, the error that we want to eliminate. Below an example is given of how the attention layer could be used to gain extra information about the model's prediction. In figures 22 and 25, the attention layer is visualized. It shows the most important hours for the final prediction. Figures 23 and 26 show how the attention layer can be used in combination with the features. Figures 24 and 27 show how our model uses SHAP to order features based on their importance. Together, these visualizations can help the decision-making process of doctors. Especially together, the information is valuable as the SHAP values stacked on top of the attention layer and development of the survival score can indicate trends in the patient's health.

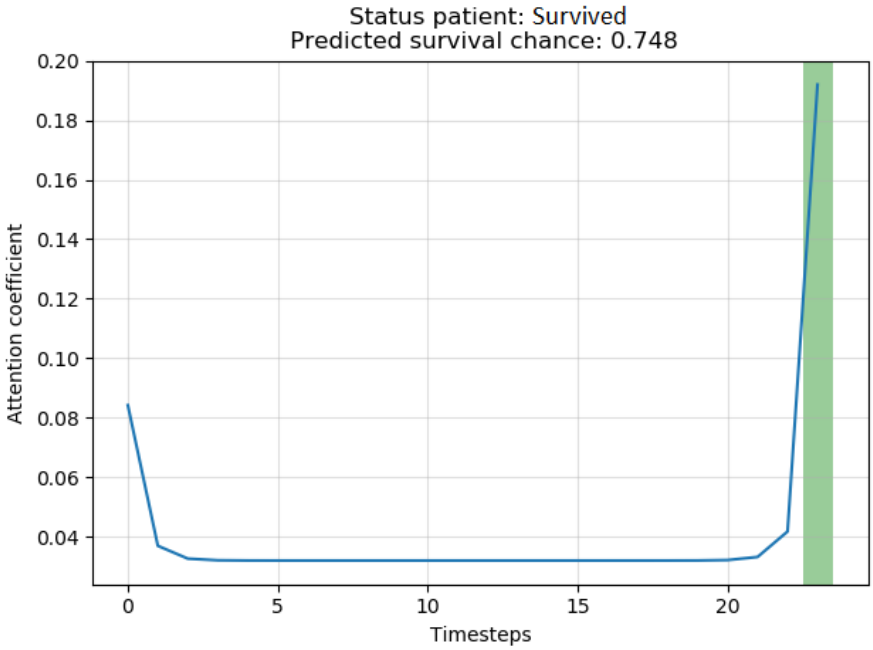


Figure 22: Attention model on a patient that survived. The data is from a random patient in the model created with the hospital 2 dataset without medicines.

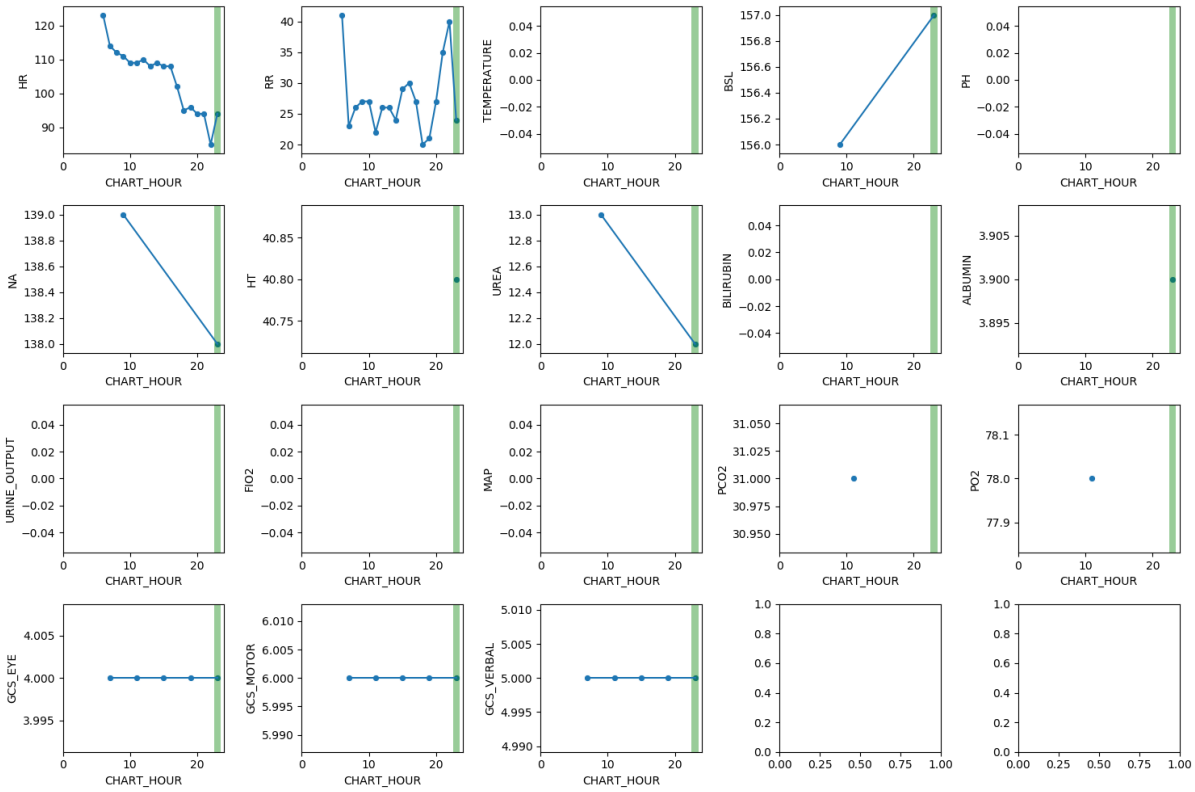


Figure 23: Attention model overlay on the features of the patient from Figure 22. The data is from a random patient in the model created with the hospital 2 dataset without medicines.

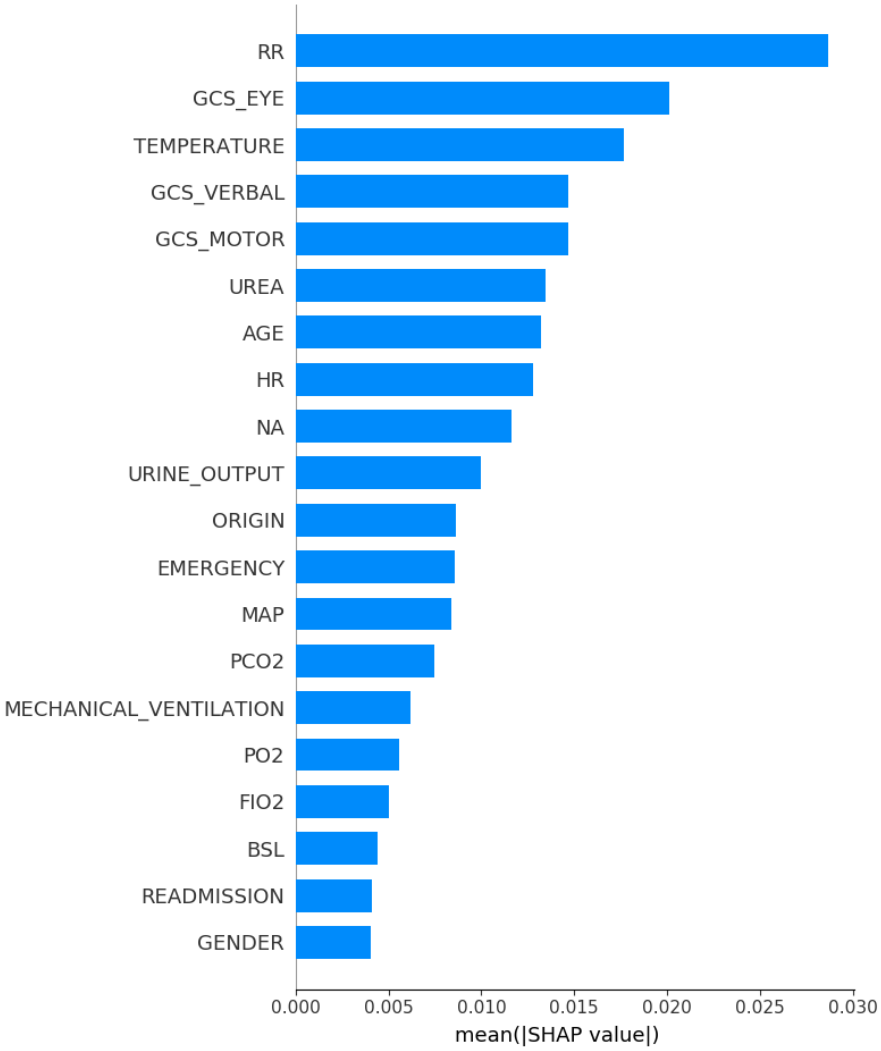


Figure 24: SHAP values ranked by importance on the features of the patient from Figure 22. The data is from a random patient in the model created with the hospital 2 dataset without medicines. On the y-axis the top 20 of most important features are visualized. On the x-axis, the impact is visualized.

At the figures above we can see how our attention model and SHAP values can provide insight into the model for a patient that survived the hospitalization. In Figure 22, the attention is plotted for a patient. The green stroke indicates the hour with the most weight in the prediction. We can see that the last hours have the highest weight. Figure 23 shows the features of the patient with the attention layer highlighted. For example, we can see that the heart rate recovers in the highlighted stroke and that the respiratory rate stabilizes. Figure 24 shows the SHAP values for this patient. It indicates the features ranked on their weight in the prediction. In this particular case, the respiratory rate, Glasgow Coma score for the eyes and the temperature are indicated as the most important.

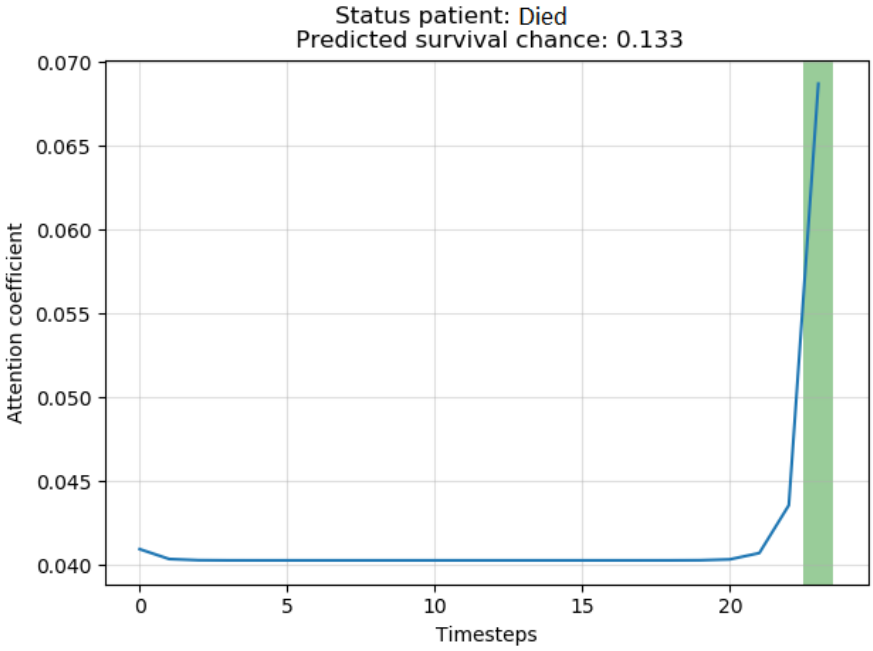


Figure 25: Attention model on a patient that died. The data is from a random patient in the model created with the hospital 2 dataset without medicines.

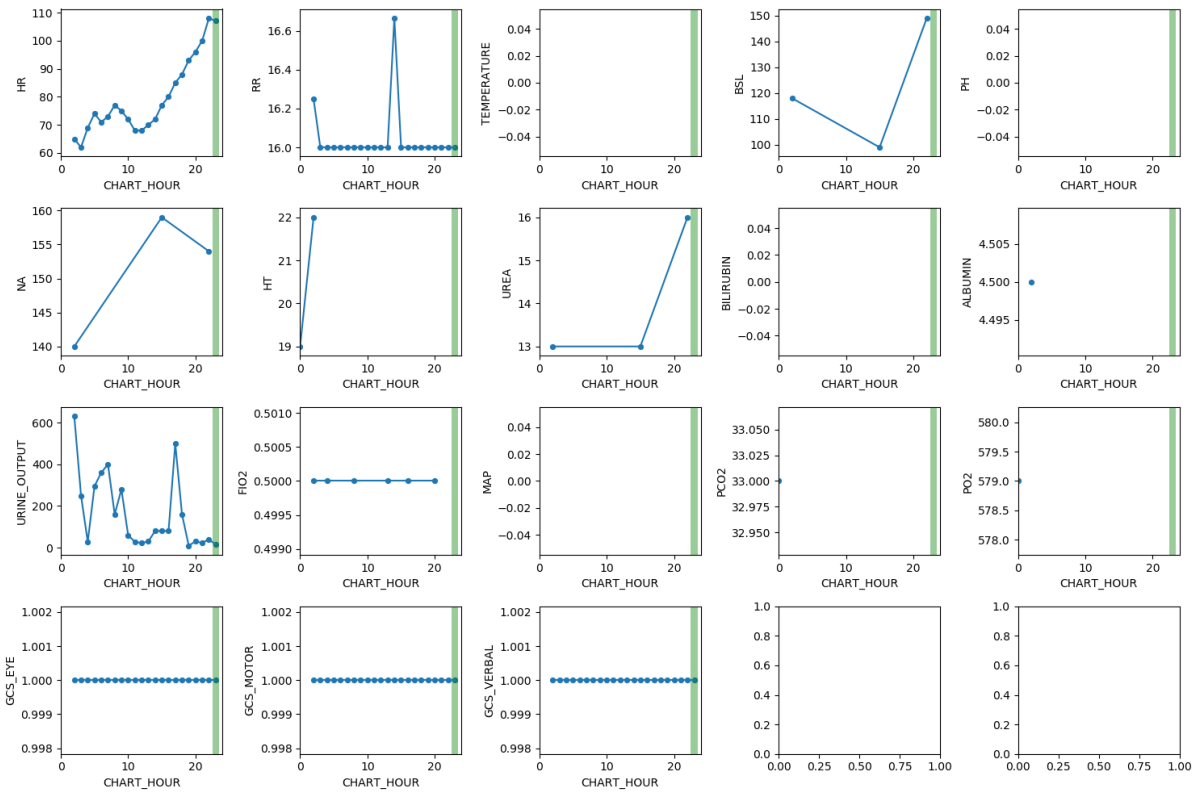


Figure 26: Attention model overlay on the features of the patient from Figure 25. The data is from a random patient in the model created with the hospital 2 dataset without medicines.

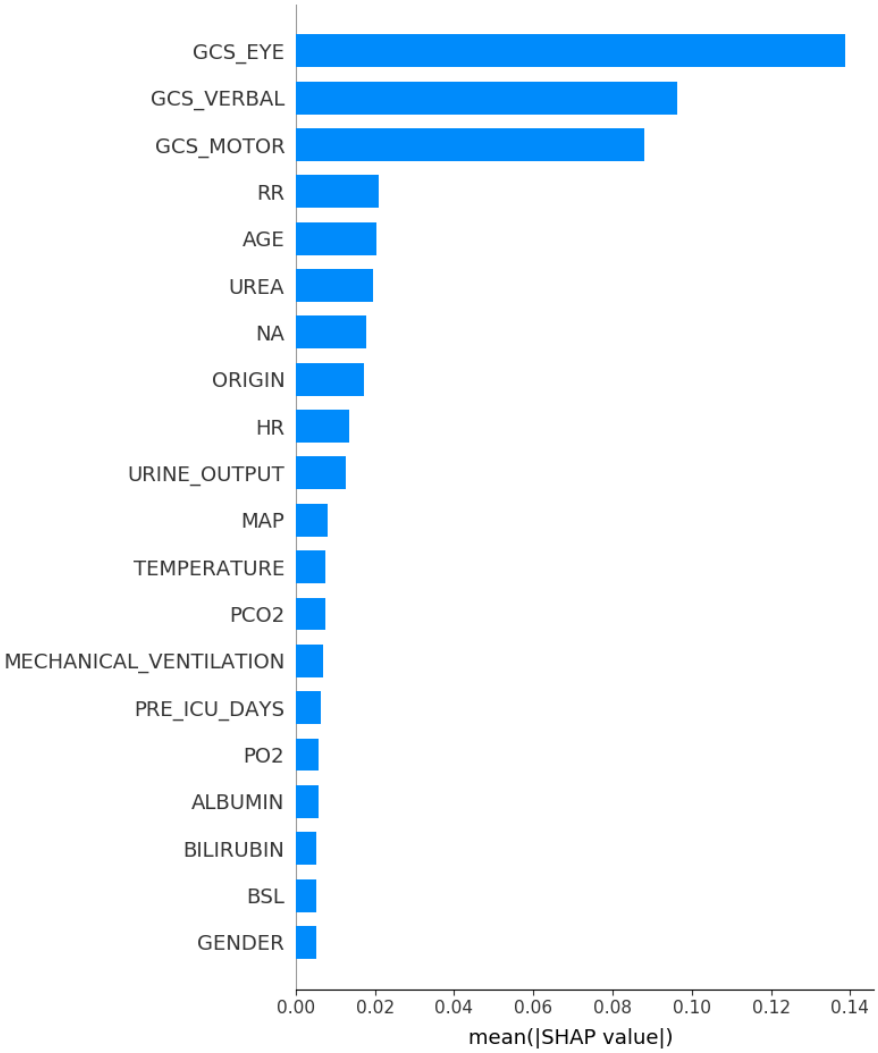


Figure 27: SHAP values ranked by importance on the features of the patient from Figure 25. The data is from a random patient in the model created with the hospital 2 dataset without medicines. On the y-axis the top 20 of most important features are visualized. On the x-axis, the impact is visualized.

The figures above indicate how our attention model and SHAP values can provide insight into the model. In this case for a patient that did not survive the hospitalization. In figure 25, the attention is plotted for a patient. We can see that again the last hours have the highest weight. Figure 26 shows us the features of the patient with the attention layer highlighted. We can see that the heart rate, urea, and BSL for example rise in the highlighted stroke. Figure 27 shows the SHAP values for this patient. In this particular case, the Glasgow Coma scores are indicated as the most important. Their values are all at the lowest possible score which indicates the person is highly comatose.

4.2 Experiment 2: 100 hours

The next experiment was about predicting more than just a single day. This is more advanced than APACHE IV and therefore there are no comparisons made between APACHE and our model. The approach for the experiments is similar to the previous experiments. First, we synthesized an optimal model with Hyperopt based on 10 evaluations. This model was then used to determine the performance metrics. In this particular experiment, it is interesting to see the AUC ROC development over time. In other words, to see how the model that is trained on 100 hours, performs on smaller subsets like 24, 48 and 72 hours. This will indicate if the model is able to perform when only a fraction of the data is available. For example, can we use this model after a patient is 24 hours on the ICU. In Tables 9 and 10 the optimal settings for each model can be found. In Table 11 the results can be found of the model with 100 hours of data.

Table 9: Optimal model settings as returned by our Hyperopt script for LSTM

Setting	Range	Hospital 1	Hospital 1 medicines	Hospital 2	Hospital 2 medicines
LSTM units	[4, 200] (integers)	120	120	100	120
Dropout on LSTM layer	[0, 1]	0.2	0.2	0.4	0.4
Use extra dense layer (boolean)	true, false	no	no	no	no
Number of units on dense layer	[4, 24] (integers)	-	-	-	-

Table 10: Optimal model settings as returned by our Hyperopt script for CNN

Setting	Range	CNN on Hospital 2
Number of convolutional filter	[60, 160] (integers)	80
Kernel size of convolutional filters	[3, 48] (integers)	24
pool size of maxpooling layer	[2, 24] (integers)	2

Table 11: Results models with 100 hours of data

Metric	Hospital 1	Hospital 1 medicines	Hospital 2	Hospital 2 medicines	CNN on Hospital 2
Accuracy	0.82	0.81	0.91	0.92	0.90
Precision	0.82	0.81	0.91	0.92	0.90
F_1 score	0.71	0.70	0.81	0.83	0.79
Sensitivity	0.92	0.92	0.96	0.97	0.96
Specificity	0.47	0.45	0.61	0.64	0.58
AUC ROC	0.85	0.84	0.94	0.93	0.92

As the results show, the CNN network proves to be competitive. We created intermediate predictions to see how well the model can be used on intermediate timesteps. For example, when we train a model on 100 hours but want a prediction after 24 hours. The following three Figures (28, 29 and 30) show the ROC score progression. This is the AUC ROC score that the model achieves after x hours of data. For the experiments, we masked the patient's data after hour x so the model could only use the available x hours of data.

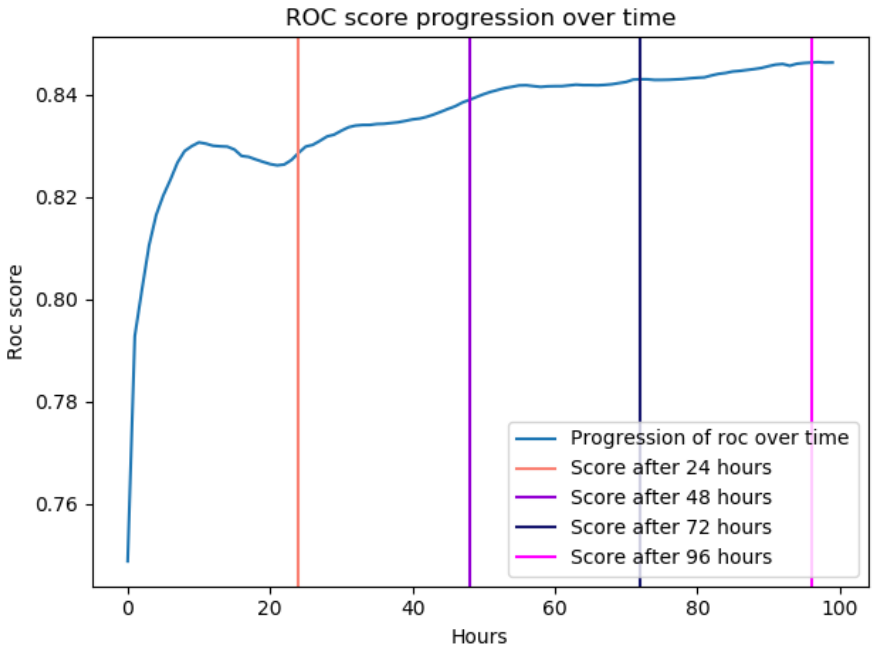


Figure 28: ROC progression of Model on the dataset of hospital 1 without medicine.

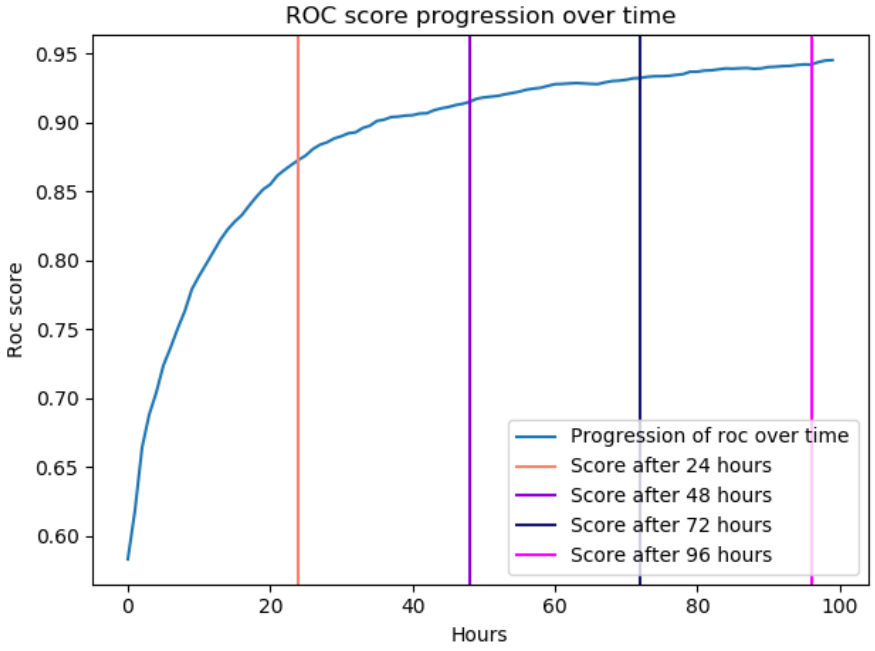


Figure 29: ROC progression of Model on the dataset of hospital 2 without medicine.

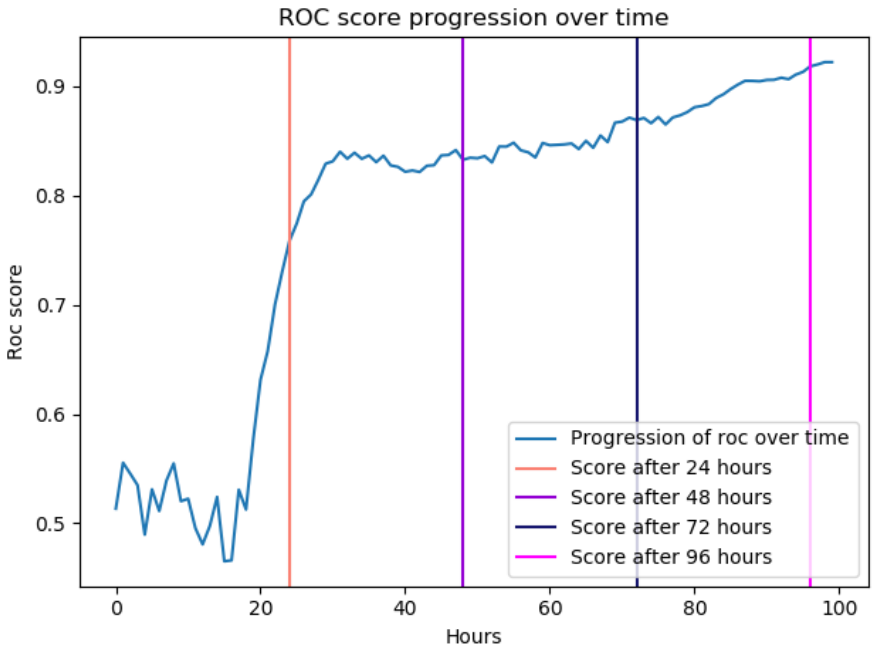


Figure 30: ROC progression of CNN on the dataset of hospital 2 without medicine.

Table 12: ROC scores after the indicated timesteps for Figures 29 and 30

Timestep	ROC for hospital 1	ROC for hospital 2	ROC for CNN on hospital 2
24 hours	0.83	0.87	0.76
48 hours	0.84	0.92	0.83
72 hours	0.84	0.93	0.86
96 hours	0.85	0.94	0.92

As is visible in Figures 28, 29 and 30, the LSTM model has a much more graduate performance and reaches an acceptable level of 0.87 AUC ROC on hospital 2 after 24 hours of data. The CNN is much more volatile in the first 24 hours and only reaches an AUC ROC score of 0.76 after 24 hours. It takes around 48 hours for the model to achieve a similar score as the LSTM model achieves in 24 hours. Although the overall score, which can be found in Table 8, does not differ that much between the LSTM and CNN, the performance over time is very different. In a real world situation where we want to use the model to make a prediction each day. A graph like the LSTM's graph is preferable as the predictions become stable after a days time. Using the LSTM model we can visualize how the prediction changes over time. In Figures 31 and 32 the predicted mortality rate is visualized over time for a patient that died and for a patient that survived.

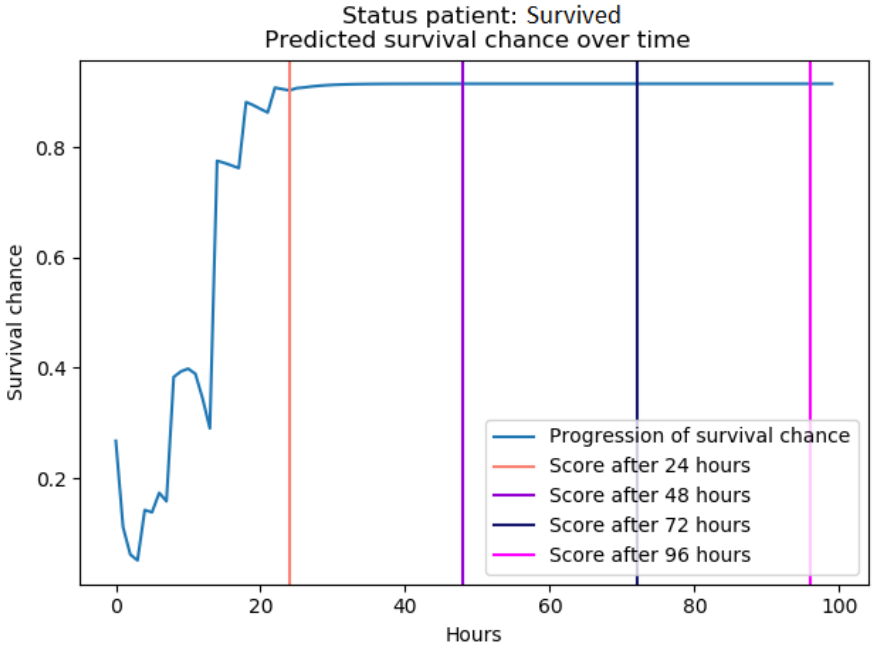


Figure 31: Development of survival prediction over time for a random patient that survived. The data is from a random patient in the model created with the dataset from hospital 2 without medicines.

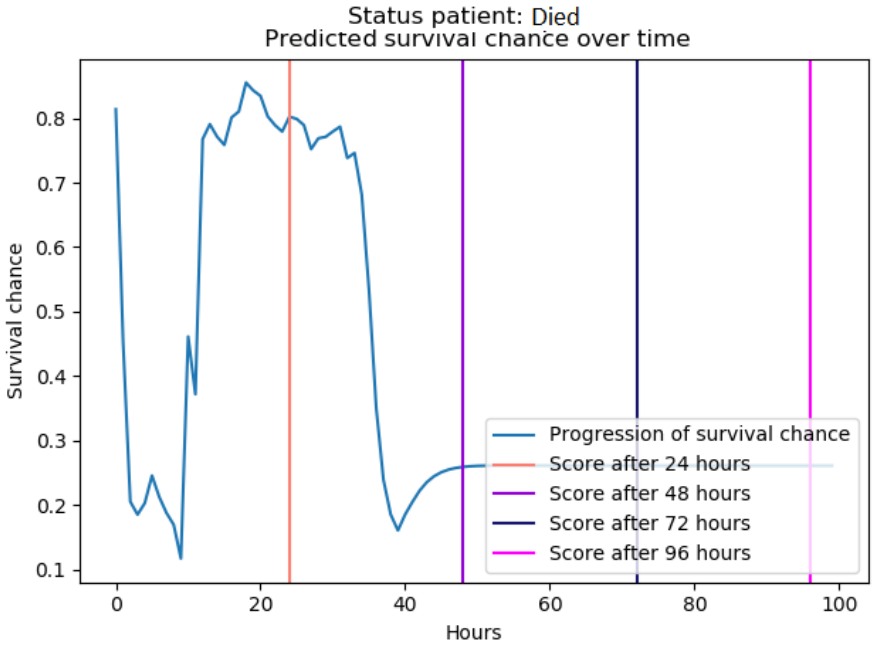


Figure 32: Development of survival prediction over time for a random patient that died. The data is from a random patient in the model created with the dataset from hospital 2 without medicines.

4.3 Experiment 3: 168 Hours

The biggest time sequence we used in the research is a sequence of 168 hours. This is an exact week after the patient's admission on the ICU. We chose to use 168 hours as 85% of all patients stay under a week on the ICU. Therefore this is a good amount of hours to use in our final model. We used the same approach as in our previous experiments. We ran Hyperopt to synthesize an optimal model and used the model for our experiments. In Table 13 the optimal model can be found and Table 14 shows the results of the model.

Table 13: Optimal model settings as returned by our Hyperopt script

Setting	Range	Hospital 1	Hospital 1 medicines	Hospital 2	Hospital 2 medicines
LSTM units	[4, 200] (integers)	120	140	100	120
Dropout on LSTM layer	[0, 1]	0.4	0.4	0.3	0.2
Use extra dense layer (boolean)	true, false	no	no	no	no
Number of units on dense layer	[4, 24] (integers)	-	-	-	-

Table 14: Results of after trained model of 168 hours

Metric	Hospital 1	Hospital 1 medicines	Hospital 2	Hospital 2 medicines
Accuracy	0.90	0.92	0.92	0.93
Precision	0.90	0.92	0.92	0.93
F1 score	0.82	0.82	0.82	0.85
Sensitivity	1	0.98	0.97	0.97
Specificity	0.53	0.58	0.63	0.69
AUC ROC	0.94	0.96	0.95	0.96

The results in Table 14 show that the model with 168 hours of data has really high performance compared to the other experiments we did. The model achieves an AUC ROC score of 0.96 on both hospital 1 and hospital 2. If we compare the score that we got with hospital 2 to the experiments with 24 hours where we had an AUC ROC of 0.89 and to the experiments with 100 hours of data where we had an AUC ROC of 0.94, we see that this is a significant rise. The AUC ROC and F_1 scores are the highest of all experiments for the dataset of hospital 2 as well as for hospital 1. The 168-hour model shows really good results but what is important is how the performance develops over time. In Figures 33 and 34 we can see the AUC ROC development over time for a model that has been trained on 168 hours of data. Both graphs show a steady rise in performance although the graph of hospital 1 takes longer to reach an acceptable score. In Table 15 we can find the precise AUC ROC value on the indicated timesteps.

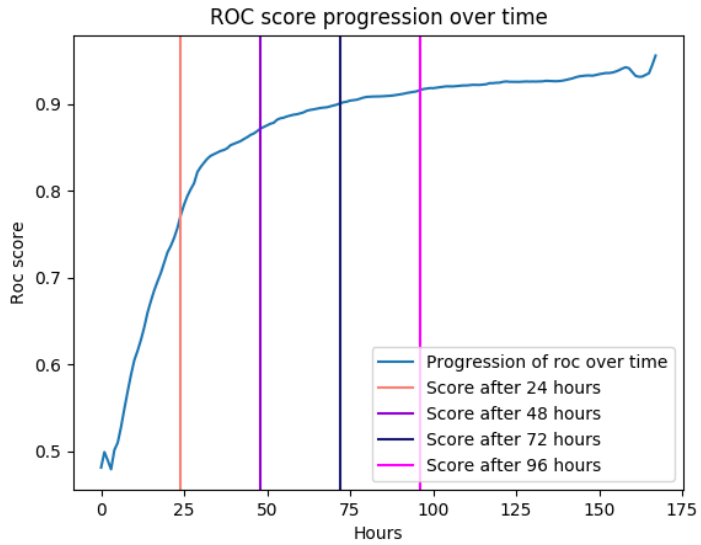


Figure 33: ROC development over time for model trained on 168 hours of hospital 1 data without medicine.

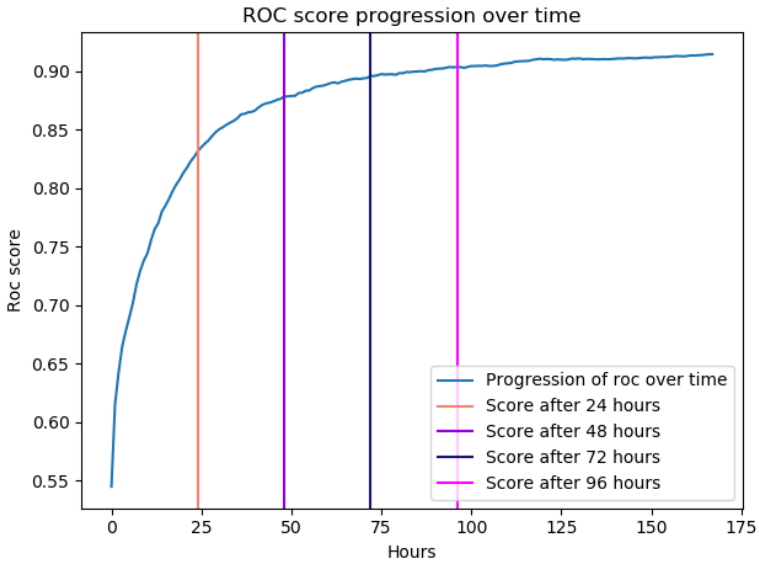


Figure 34: ROC development over time for model trained on 168 hours of hospital 2 data without medicine.

Table 15: ROC scores after the indicated timesteps for Figures 34 and 33

Timestep	ROC on hospital 1	ROC on hospital 2
24 hours	0.78	0.83
48 hours	0.85	0.87
72 hours	0.88	0.89
96 hours	0.91	0.90

5 Implementation

In the previous chapter we've shown the performance of our model but the success of our model depends on the implementation of it. To prove that our model is of added value, we need data to evaluate this. Therefore it is crucial that doctors will use our model. To achieve a high user participation, the model should be easy to use, accessible and it should be beneficial to use for the doctor. To accommodate these requirements, we've built a dashboard where we embedded our model in. This dashboard is accessible for ICU employees and can be consulted by the staff. As it requires a maintainable software stack, we used a microservice architecture to deploy our software [35]. The dashboard itself is build using Vuejs [29]. This is a Javascript framework. We've hosted the website internally so it is only available from within the LUMC. The framework makes API calls to our own Flask API [30]. Again the Flask API is hosted inside the LUMC but as a different service. This service only hosts APIs for our clients. For example, when the dashboard page is started, the data is fetched from the 'GET patients' API. The API gets the patient data from the database. This database already existed and is hosted on the 'Dataplatform'. This is an internal project in the LUMC to combine all their databases into a single service. The patient data is then processed and fed to our model, which adds the predictions. The result is posted to the client where the data is displayed on the dashboard. Our AI model is also hosted as a separate service. We chose to use a microservice architecture as this allows us to quickly make changes in the production software, this makes the app more scalable and improves the availability. In Figure 35 a simplified diagram with the architecture of our app is shown.

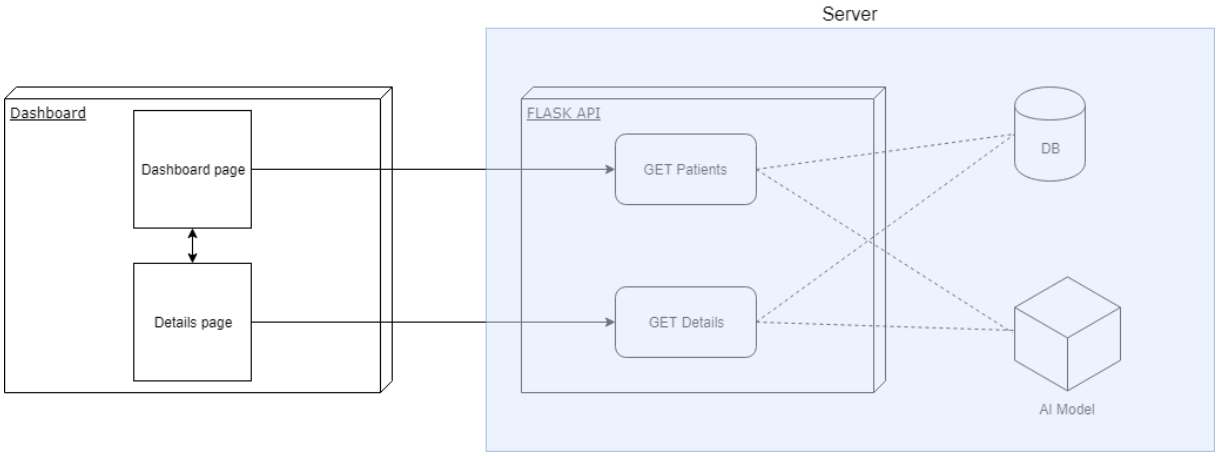


Figure 35: Software Architecture diagram. This is a simplified model. In reality the Flask app has more APIs and connections to the dashboard, AI model and database.

We've kept the app as simple as possible to reduce the amount of maintenance required. Therefore the app has only two pages. The landing page of the dashboard shows the bed number, name, length of stay and the model prediction for all current patients. In total, we created three different views. The dashboard contains a table view, a card view, and a map view. Each view has a different purpose and can be used in different situations. The table view is the main view. It can be sorted by clicking on the table headers and gives a clear view of the patients. The second view is a card view that can be sorted and filtered on the different properties. The last view is called the map view. This view is based on the current system the doctors use to browse the patient data. The view shows the patients per department and allows for sorting and filtering.
Along with the patient data, some general data about the population of patients is displayed here. The view supports all kinds of querying and sorting options. In Figure 36 an impression of the dashboard can be found.

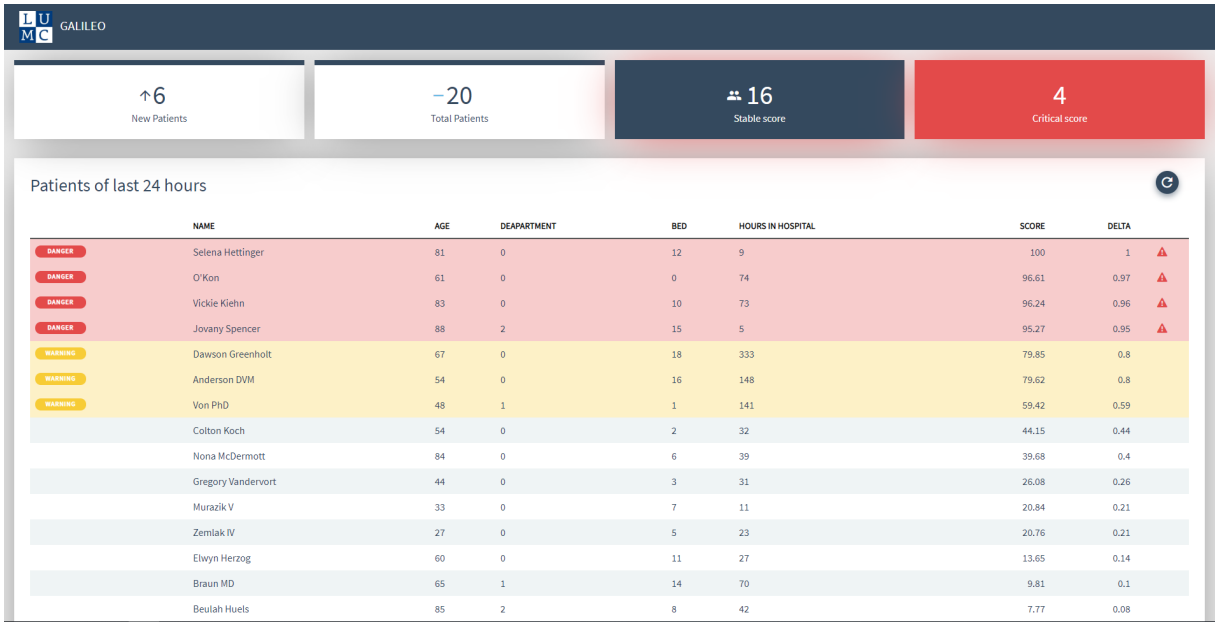


Figure 36: Landing page of dashboard. The data on the dashboard is faked.

Clicking on a patient will open a detail page. Here all the details of the patient can be found. The screen opens with some static data of the patient, the historic course of the mortality prediction, the importance of each feature, an overview of all feature measurements over time and a timeline of procedures. Clicking on a feature opens a dialog that visualizes the feature over time. On the bottom of the page, the overview of all features and the procedure timeline can be found. Clicking on the procedure timeline shows all procedures that are executed over time. Clicking on the overview of all features will expand the page with graphs of all features over time. Figure 37 shows an impression of the details page.

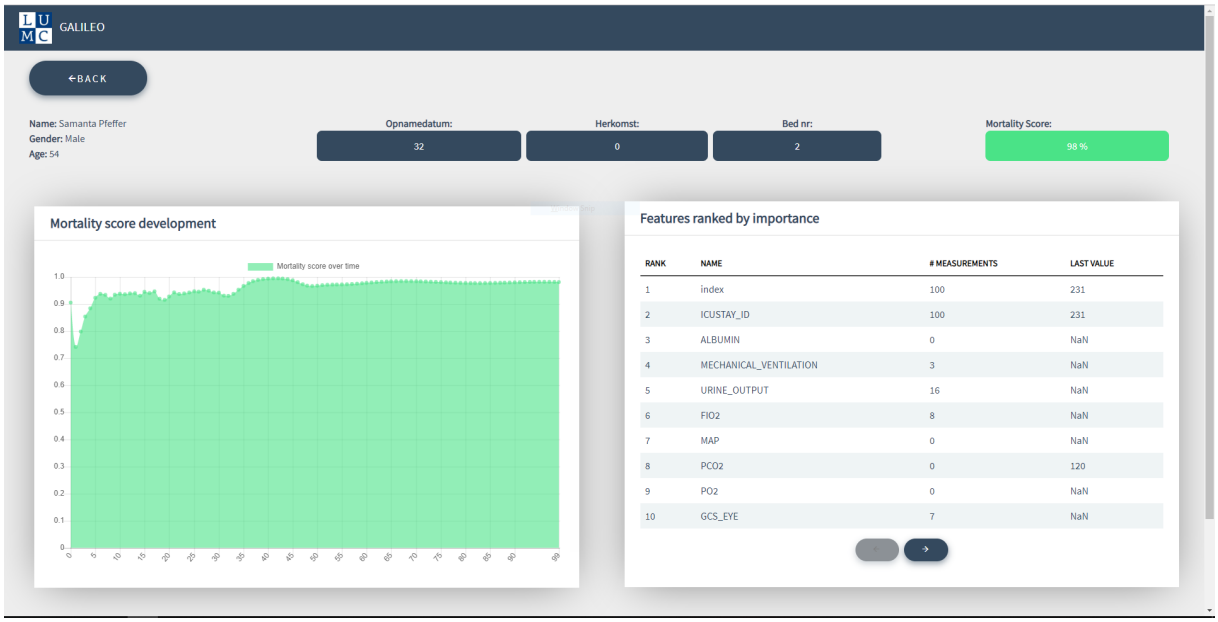


Figure 37: Detail page of dashboard. The data on the dashboard is faked.

6 Discussion of results

In this chapter, we will discuss the results of our research. The experiments are discussed as well as the implementation chapter.

6.1 24 Hours

If we take a look at Table 7, the results of the first experiments, we can see that the AI model greatly outperforms the conventional model. Our model on hospital 2 achieves an AUC ROC score of 0.89, the model on hospital 1 achieves a score of 0.83 and the APACHE IV predictions only scores 0.74 AUC ROC. The APACHE scores we've used for the research are real scores for the same patients we've used in our model. This gives an even better view of the results and shows the superior results of the AI model. The better performance can be explained by the amount of data that the AI model has access to compared to the data used in APACHE. APACHE IV only uses the worst values for each feature in the last 24 hours. This means that it only uses the worst heart rate, respiratory rate, PH, etc. Our model uses the mean values of each hour. If we look at a hospital stay and in particular an ICU stay, a lot can happen in 24 hours. For example, when a patient that is transferred to the ICU is in a really bad shape early on, APACHE will use those values. If the patient recovers after some procedures, this is ignored by APACHE. However, our model does use all this data and can, therefore, give a more accurate prediction. One big difference that comes with the added data for our model is that this includes the clinician's procedures and treatments for the patient. Hypothetically a patient could be in a really bad state but due to the intensive treatment, he/she is kept stable. Our model could predict a high survival chance as the patient looks stable. On its own, this is not a problem, as the model has a good performance. It is, however, interesting to see how the doctor's attention and perception of the patients are influenced by this. Measuring this bias and the impact on the health care for the patients will be crucial for the success of our model.

Another striking result is the difference in performance between the models from hospital 1 and hospital 2. Same as with the difference in mortality rate, it is hard to pinpoint where the exact difference comes from. One factor is explained in chapter 2. There are big differences in the health care systems that are active in both countries. In the US not everyone has access to or can afford health care and especially health care in a teaching or university hospital. In The Netherlands, more people have access to health care [32]. Another factor is the amount of training data. The dataset from hospital 2 has two times more data than the dataset from hospital 1. The amount of training data is crucial for a good model. In our case where hospital 2 has twice as many patients, the model has an easier time adapting to the data and finding patterns. In hospital 1 there are fewer patients and examples to learn from.

When we look at the addition of medicine we see that the added features result in a better performance of the model. Both for hospital 1 and for hospital 2 the AUC ROC scores increase with the added features. It is however strange that hospital 1 has a lower F_1 score. We can see that the precision is also lower but the sensitivity is higher, thus the model has more false negatives but less false positives. As we trained our model using a cost function, penalizing false positives more than false negatives, this result is still an improvement over the result without medicine. A difference between both is that hospital 2 had a categorization of 88 different drug types while hospital 1 has only 28 categories. The categorization of hospital 2 is an international standard whereas the categorization in hospital 1 is an internal categorization. The international categorization could be more descriptive than the internal one.

The confusion matrices in Figures 19, 20 and 21 show that the class weights have helped our model to minimize the false positives. As described in chapters 3.3 and 3.4, we wanted to minimize the number of patients with high predicted survival chances that actually die. Compared to APACHE's confusion matrix, we can say that we achieved this.

The attention layers, visualized in Figures 22 and 25, show the important hours in the final prediction. In both figures the last hours of the patients first 24 hours seemed to be most important. In Figures 23 and 26 we've put the attention layer over the features to visualize the important trends. For the surviving patient, we can see that the respiratory rate dropped, the temperature rose and the MAP dropped. For the patient who did not survive the ICU stay, the respiratory rate rose, the PO2 and the PCO2 dropped. In Figures 24 and 27, the SHAP values can be found. For the patient that survived, the respiratory rate and the Glasgow Coma Scores (GCS) were important. For the patient that did not survive, the Glasgow coma scores were important. If we look at the values in Figure 24 we can see that the GCS scores are 1 at all hours. This means that the patient did not responded to the tests. For a doctor these are indicators of a patient's status.

6.2 100 Hours

The results of Table 11 show an improved model. The extra hours of data dramatically improved the model. The ROC and F_1 scores of both models have been improved. Again hospital 2 manages to get better results than hospital 1. Same as for the 24-hour experiment, the same reasons count here as well.
The CNN experiment we did showed very good results in Table 11. The performance is only slightly worse than the LSTM model. As this CNN model was very shallow and could easily be improved by tuning with extra layers, it was a very good result. Additionally the model trained really fast, allowing us to better tweak it to an optimal configuration.
When we started this research, one of our goals was to develop a model that was able to predict continuous timesteps. In an ideal scenario, we would have one single model that is able to predict patient mortality after 24, 48 and 72 (and onwards) hours. The scores in Table 11 only show the performance after 100 hours and are not representative for the performance of the model when it only has a subset of 100 hours of data. Therefore we plotted the AUC ROC development over time. We've calculated 100 different AUC ROC scores for the test set, for each hour once. For hour 1 we only kept the first hour of data and masked the rest, for hour 25 we only kept the first 25 hours of data. This gave us the plots displayed in Figures 29 and 30.
Here it is clear that the LSTM, which is displayed in Figure 29 learns at a quicker and more steady rate than the CNN. After 24 hours of data, the results are comparable to the results in our first 24 hour experiments. This meant that a single model could be used to predict at varying timesteps. The CNN, visualized in Figure 30, shows a less steady curve, only overachieving the 24-hour experiments of the LSTM after 60 hours. The performance of the CNN at intermediate steps can be explained by the lack of masking. This means that it is harder for the model to ignore the unavailable hours. For our implementation, this would mean that we should create a separate model for each timestep that we want to use our model.
In Figures 31 and 32 we've displayed how our model's judgment of a patient varies over time. In Figure 31 a surviving patient is displayed. In the first hour of the ICU stay, the patient gets a really low chance of surviving from our model. The survival chance increases rapidly and after 18 hours the patient is given a bigger chance to survive than to die. In Figure 32 we can see a patient that died during it's ICU stay. After 38 hours a big decline sets in and the predicted survival chance drops.

6.3 168 hours

The results for 100 hours were promising. We selected 168 hours as this is a full week and 85% of the ICU patients stay under a week on the ICU. In other words, we can serve the vast majority of patients with this model. The performance of the model, found in Table 14, shows that it increased compared to the results of 100 hours. This could be expected as we have more data and for 85% percent of the patient population, we have all available data.
The interesting part of this experiment is the ROC development curve. In Figures 33 and 34, the ROC development over time is visualized. We can see that this line beats the scores of the experiments with 24 and 100 hours of data. If we look at the model that was trained on hospital 2, we see that after 24 hours the model achieves a score of 0.88. This is similar to the score of 100 hours and a little bit worse than the model that is only trained on 24 hours. After 24 hours it keeps improving at a steady rate until it reaches an AUC ROC score of 0.96 after 168 hours.
The performance of the model with 168 hours of data from hospital 1 is somewhat surprising as it has almost similar metric scores as the model on hospital 2. In previous tests with 24 and 100 hours, the model of hospital 1 was significantly worse. If we look at the model's AUC ROC development over time, we see that after 24 hours, the model reaches an AUC ROC score of 0.78. After 48 hours this has risen to a score of 0.87 and eventually it will rise up to an AUC ROC score of 0.94 for a dataset without medicine.
The results are promising and indicate that a model that is trained on 168 hours of data, is able to make accurate predictions on various timesteps. For example, we could use the model that is trained on 168 hours of data to predict a patient's score after 24, 48 or 72 hours. The results from Table 15 show that the loss of accuracy is minimal. After 24 hours the 168-hour model has an AUC ROC score of 0.78 whereas a model that is solely trained on the first 24 hours of data (Table 8) achieves a score of 0.84. If we compare this to our 100-hour model, where we had a AUC ROC score of 0.83 after 24 hours, we can see that there is trade-off between the 100- and 168-hour model. The 100-hour model has an acceptable score all the time whereas the 168-hour model has a really good score at the end and can serve a bigger portion of patients but scores low after 24 hours. As the model will be compared to APACHE IV, it is better to choose a model that is more steady after 24 hours.

6.4 Implementation

As mentioned in chapter 5, the success of the model depends on the implementation. In other words, if the model is not used, we won't help the doctor nor patient with our model. We therefore put a decent amount of time in the realization of a dashboard. The dashboard we've created can be used to test and validate our model. The dashboard is developed as a simple and clear overview of the patient's status and the model's prediction. The simple nature of the architecture and the application make it easy to maintain or extend it. An infrastructure is developed that enables other members of the team to further develop the application.
The microservice architecture separates the various components by their responsibilities and allows for more scalability, availability and a better maintainability. The services can each be deployed separately and are orchestrated to work together.
The dashboard overview can be used by the doctors during their daily transmission where they discuss each patient. The overview offers a clear and simple overview of the current state of each patient. Clicking on the patient will open the detail page, where the doctors have an overview of how the model reasons and can see the details of the patient's current and historic status.
On this detail page, there is the option to zoom in on the patients ICU stay. It provides a way of visualizing the patient's measurements over time. It also contains a timeline of the treatments that a patient received during his/her stay.
As we developed our dashboard with a high priority on maintainability, it is possible to add new components to the dashboard without too much work. For example, when an additional model is developed this can be added in the dashboard simply by creating an API and visualizing it.

7 Conclusion

Before we started, we defined three goals. Our first goal was to outperform a conventional model like APACHE IV. If we look at the results chapter we can clearly see that our model outperforms APACHE IV. The F_1 and AUC ROC scores are both a lot higher and the confusion matrices show that our model has relatively less false positives. The APACHE scores we've used for the research are real scores for the same patients we've used in our model. This gives an even better view of the results and shows the superior results of the AI model. As described in the discussion, the model does use different data to make its predictions. A clinician who consults the model should be aware of this. After all a patient who is predicted to have a high chance of survival is still admitted to the ICU.

Our second goal was to develop a model that is capable of predicting patient mortality on a continuous scale. This has been achieved. A model that is capable of predicting after 100 or 168 hours still has an acceptable performance after 24 hours. This means that the model can be used on an hourly basis to predict the patient's mortality risk. The performance of the model increases over time, which is expected as there is more data available for the model. In our case there is a trade-off between a better prediction early on against a better prediction in the end. As our model will be compared to APACHE, it is better to use a model that is accurate after 24 hours. We therefore, chose to use the model that is trained on 100 hours of data.

The last goal was to make our model descriptive and explainable. With the addition of the attention layer, we are able to pinpoint the exact hour that was decisive in the final prediction. With the SHAP values, we can order the features based on their importance. Together with the dashboard, we created for the doctors this gives a compact, conceivable and supportive view of the patient's status. The model's score is used as a categorical factor where patients are categorized by the chance of mortality. This helps clinicians to better focus their attention. The dashboard also shows the static information of a patient, the current status of the patient and the development of features and score over time. All of these data were spread and hard to access. The model we've created summarizes all these data and our dashboard offers the doctors an accessible way of evaluating a patient's status.

The results of our research will help doctors to improve health for critically ill patients. The model will help them focus their attention better through the immense forests of unstructured data. Our dashboard also supports them in the decision-making process and has the ability to report a summary of the patient's health status. We believe that the vast amounts of data can and should be used for the patient's well being. With this research, we are one step closer to this goal.

8 Future work

Future projects that expand upon this research could improve the performance of the model by adding new features. This would give the model more information to find and learn relationships in data. A possibly interesting group of features to add are procedures. For example, if a patient is connected to an ECMO [36]. In both hospital 1 and hospital 2, there are ICD-9 codes that can be categorized into different diseases and procedures [37]. The extra features could improve the model's performance substantially. Other research also used procedures [4] and [9].

Additionally new research could focus on predicting the length of stay. A similar architecture could be used to achieve this. The predicted length of stay can be used to estimate the capacity and can help focus on health care. The currently used mortality prediction model, APACHE IV, also predicts the length of stay. So improving on that score would make it possible to remove APACHE IV from the daily routine and to only use the custom model. Other research also predicted the length of stay [4] and [9] and used a similar approach as they did when predicting the mortality.

Another interesting extension would be to further develop our dashboard and adjust it even more to the wishes of the doctors. Possible extensions would be to add more customization on the graphs like the possibility to zoom into specific times and adding more features that are not used in the model. This will allow doctors to use them in the decision-making process. Another possible extension would be to integrate new models and predictions that support the clinicians.

Very interesting follow-up research would be to validate and test the implemented model. This was not possible for this research as there were time constraints but it is really interesting. In this research, we've validated our model on historical data. This gave us very promising results and we, therefore, started developing the model into a full implementation. Given our results, we can say that our model performs better than APACHE and that it performs very well over a longer time. Our model makes predictions but is in no case the truth. Next to this, the model predicts a patient's mortality using historic and future treatments. This means that when a patient has a low risk of mortality and the doctor stops treating the patient, he/she can still die. On the other hand, a patient with a high mortality risk can be saved when given the proper treatment. Figures 31 and 32 give a good representation of these cases. In Figure 31, a patient who survived the ICU stay, had a very bad score early on but recovered during the next hours. In Figure 32, the patient recovered after the first few hours but still died during the hospital stay after a sudden decline on the second day. The above questions make clear that although our model performs well, it is extremely important to see how doctors use the model and to verify our model on actual patients that are on the ICU. A big question is where the model should be evaluated on. One option is to evaluate the model on changed mortality in the hospital. It is questionable if this model will reduce patient mortality at all as the health care that patients receive will probably not change. Another option is to ask clinicians how it affects their work. For example, if they can manage their attention better and have an increased focus for high-risk mortality patients, this could be beneficial for the patients.

To validate the model on these patients we should be really careful not to make biased assumptions. We, therefore, developed a four-step validation and testing approach.

The first step in validating the model should be to validate it behind the scenes. This will make it possible to validate the model on real patients without introducing bias. We would expect to have similar results as in our experiments. The model's results and information can be presented to a doctor who can verify the model and give feedback. An important part of this step is to see if the clinician can use the descriptive function of our model and if it makes sense. In other words, given patient x dying on time t, can our model predict the patient's dead before t and can the doctor use the model to act.

In step 2 the model should be tested in the clinic. The preferable method would be A/B testing. This means that one-half of the ICU patients and clinicians get to use the new dashboard and model and the other half doesn't. This way the result of a model can be tested with a test and control group. It also allows us to compare the results with the results from step 1 to see if the doctors develop a bias with the support of the model. This stage is crucial for the acceptance and validation of the new model.

In the third step, we should update the model. This can be done after step 2 in the first iteration or after step 4. This step is used to retrain our model on the biases of the doctors. When we use our model, the doctors will probably behave differently as opposed to not using the model. This behavior can be measured in the treatment or care they give to a patient. The model should be updated with the latest examples. This step should be repeated every once in a while.

In the last step, we can implement our model fully for every patient and doctor on the ICU. The model has been verified and tested in real cases and the model is updated to handle biases. It is only in this step that we can measure the full impact our model makes in the health care and which benefits it brings for the patients.

9 Acknowledgements

I would like to thank some people for their contribution to this project. First I would like to thank Dr. Kaifeng Yang who supported me throughout this and my previous project. It is thanks to him that I could successfully execute this research. I am also really grateful that I got the opportunity to combine this research with an internship at the LUMC where I got great support from Simone Cammel and Esmee Stoop and had a chance to learn in a professional environment. Other thanks go to Dr. W.J. Kowalczyk for the useful discussion we had, Dr. Sesmu Arbous for her advice and cooperation on this project, my parents and my girlfriend for their encouraging words and support and the friends that I made during my time at the LIACS.

Reproducibility

The scripts that are used in the experiments of this report are in a secured Gitlab [38] environment that is hosted and controlled by the LUMC. Access to the repository can be requested by emailing to digitaleinnovatie@lumc.nl. The LUMC data used in this report is confidential and cannot be used. The MIMIC data however can be requested on the official website [3]. To gain access to the MIMIC dataset, one has to complete an online course.

References

[1] S. Kim, W. Kim, and R. W. Park, *"A comparison of intensive care unit mortality prediction models through the use of data mining techniques"*, Healthcare informatics research, vol. 17, no. 4, pp. 232–243, 2011.

[2] S. Lemeshow and J.-R. Le, *"Modeling the severity of illness of icu patients: a systems update"*, Jama, vol. 272, no. 13, pp. 1049–1055, 1994

[3] MIMIC-III, a freely accessible critical care database. Johnson AEW, Pollard TJ, Shen L, Lehman L, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, and Mark RG. Scientific Data (2016). DOI: 10.1038/sdata.2016.35. Available from: http://www.nature.com/articles/sdata201635

[4] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun et al., *"Scalable and accurate deep learning with electronic health records"*, npj Digital Medicine, vol. 1, no. 1, p. 18, 2018.

[5] Khan Academy, "Identifying outliers with the 1.5xIQR rule" accessed March 2019. *https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/identifying-outliers-iqr-rule*

[6] Urvashi Jaitley on Medium, "Why Data Normalization is necessary for Machine Learning models", 7 October 2018. *https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029*

[7] R. Pascanu, T. Mikolov, and Y. Bengio, *"On the difficulty of training recurrent neural networks"*, in International Conference on Machine Learning, 2013, pp. 1310–1318.

[8] J. Chung, C. Gulcehre, K. Cho,Y. Bengio. *"Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling"*, cs.NE 11 Dec 2014(arXiv:1412.3555 [cs.NE]).

[9] M. Aczon, D. Ledbetter, L. Ho, A. Gunny, A. Flynn, J. Williams, and R. Wetzel, *"Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks"*, arXiv preprint arXiv:1701.06675, 2017

[10] *"A tour of recurrent neural network algorithms for deep learning"*, Jul 2017. [Online]. *https://machinelearningmastery.com/recurrent-neural-network-algorithms-for-deep-learning/*

[11] Christopher Olah, *"Understanding LSTM Networks*, August 27, 2015. Available on: *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

[12] Keras library, Chollet, Francois and others, 2015. *https://keras.io/*

[13] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[14] Skymind, *"A Beginner's Guide to Attention Mechanisms and Memory Networks"*. accessed March 2019, *https://skymind.ai/wiki/attention-mechanism-memory-network*

[15] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, *"Neuroscience-inspired artificial intelligence"*, Neuron, vol. 95, no. 2, pp. 245–258, 2017.

[16] Github, *"LzFelix Github repository"*. Available: *https://github.com/lzfelix/keras*$_{attention}$

[17] Bergstra, J., Yamins, D., Cox, D. D. (2013) Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. To appear in Proc. of the 30th International Conference on Machine Learning (ICML 2013).

[18] Amar Budhiraja, "Dropout in (Deep) Machine learning". Dec 15, 2016. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[19] J. E. Zimmerman, A. A. Kramer, D. S. McNair, and F. M. Malila, "Acute physiology and chronic health evaluation (apache) iv: hospital mortality assessment for todays critically ill patients" Critical care medicine, vol. 34, no. 5, pp. 1297–1310, 2006.

[20] Michael Avendi, PhD, Medium. "Another look into overfitting", March 1, 2018. https://medium.com/randomai/another-look-into-over-fitting-33e15b044a5e

[21] Sem Spirit, Receiver Operating Characteristic (ROC) Curves, 2016. http://www.semspirit.com/artificial-intelligence/machine-learning/classification/classifier-evaluation/receiver-operating-characteristic-roc-curves/

[22] A. Borovykh, S. Bohte, C.W. Oosterlee, "Conditional time series forecasting with convolutional neural networks", arXiv:1703.04691v5 [stat.ML] 17 Sep 2018.

[23] INTERNATIONAL CLASSIFICATION OF DISEASES, NINTH REVISION ICD-9 PDF https://simba.isr.umich.edu/restricted/docs/Mortality/icd$_0$9$_c$$odes$.$pdf$

[24] A. E. Johnson, T. J. Pollard, and R. G. Mark, "Reproducibility in critical care: a mortality prediction case study", in Machine Learning for Healthcare Conference, 2017, pp. 361–376.

[25] Sumit Saha, A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way, Dec 15, 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[26] GOOGLEE AI blog, A. Mordvintsev, C. Olah and M. Tyka. June 17, 2015. https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[27] Sarang Narkhede, Understanding Confusion Matrix. May 9, 2018. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

[28] Varghese, Y. E., Kalaiselvan, M. S., Renuka, M. K., Arunkumar, A. S. (2017). Comparison of acute physiology and chronic health evaluation II (APACHE II) and acute physiology and chronic health evaluation IV (APACHE IV) severity of illness scoring systems, in a multidisciplinary ICU. Journal of anaesthesiology, clinical pharmacology, 33(2), 248–253. doi:10.4103/0970-9185.209741

[29] VUEJS framework, Evan You. https://www.vuejs.org

[30] Flask, Pallets Team. 2010. http://flask.pocoo.org/docs/1.0/

[31] S. M. Lundberg, S. Lee. A Unified Approach to Interpreting Model Predictions. Part of Advances in Neural Information Processing Systems 30 (NIPS 2017). http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[32] P.C. Michaud, D. Goldman, D. Lakdawalla, A. Gailey, and Y. Zheng, Differences in Health between Americans and Western Europeans: Effects on Longevity and Public Finance. Soc Sci Med. 2011 Jul; 73(2): 254–263. Published online 2011 Jun 2.

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

[34] Hand, David. "A note on using the F-measure for evaluating record linkage algorithms - Dimensions"., app.dimensions.ai. Retrieved 2018-12-08.

[35] Chris Richardson , Microservices, 2018. (https://microservices.io/)

[36] ECMO explanation by the LUMC, LUMC, 2018. https://www.lumc.nl/org/intensive-care/patientenzorg/ECL/ECMOuitleg/

[37] Centers for Disease Control and Infection, International Classification of Diseases,Ninth Revision (ICD-9). November 6, 2015. https://www.cdc.gov/nchs/icd/icd9.htm

[38] Gitlab, visited June 21st 2019. https://about.gitlab.com/

Appendix

Appendix 1

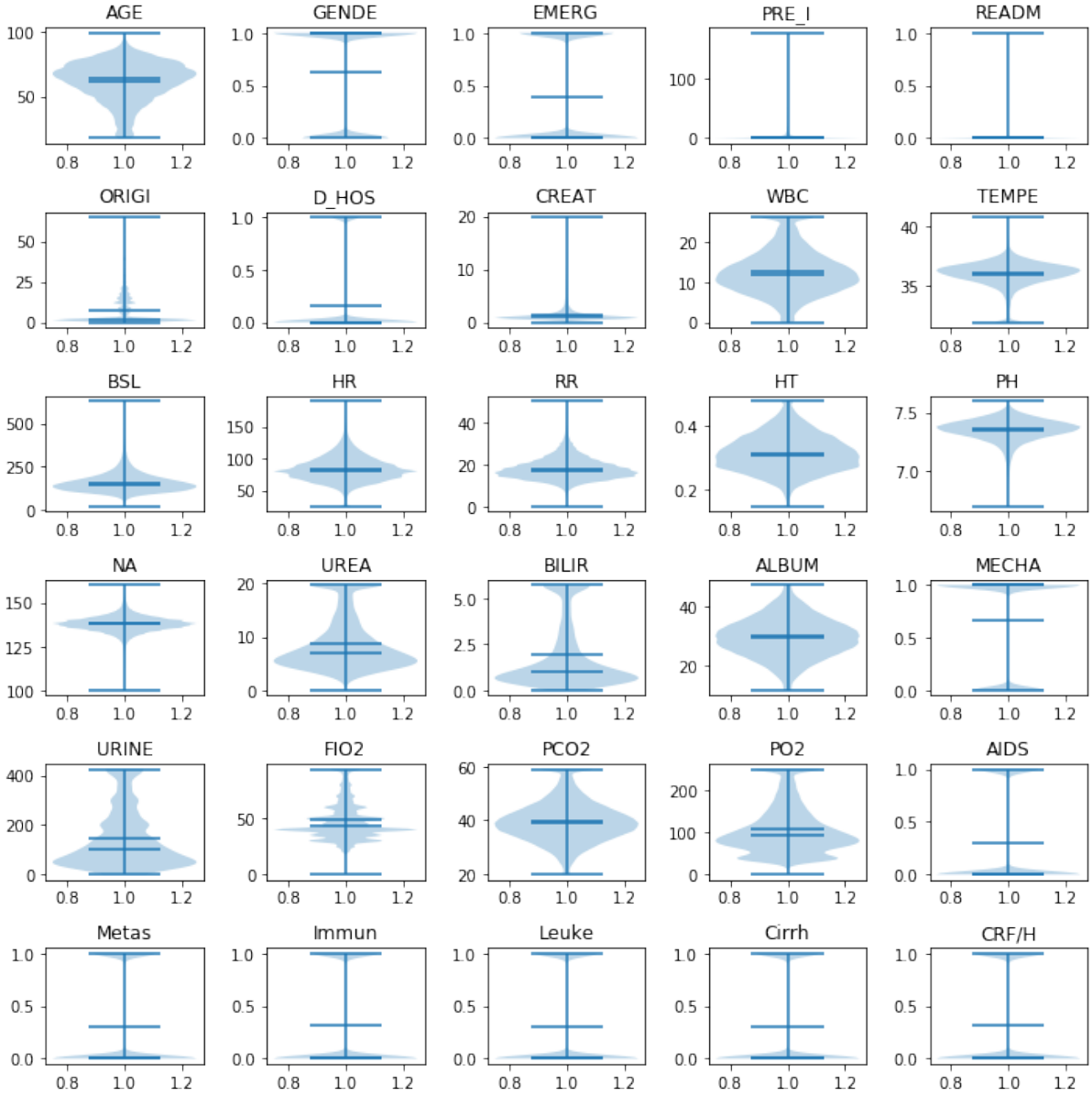


Figure A1: Violinplots of the preprocessed hospital 1 dataset. Note that this data is not normalized. Each violinplot represents a feature. The title of each plot is constructed out of the first five letters of the feature name. Please check Table 3 for the full feature names.

Appendix 2

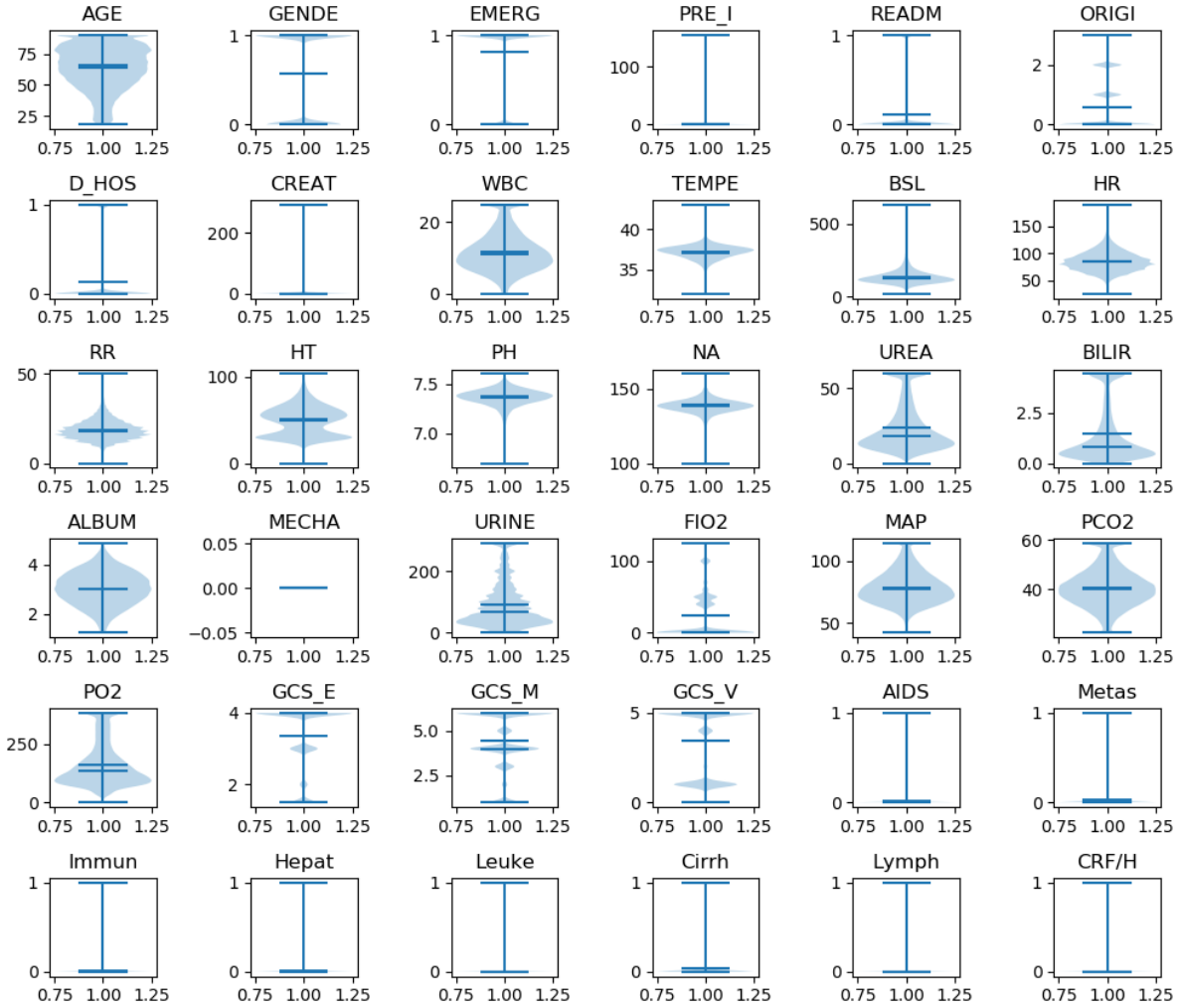


Figure A2: Violinplots of the preprocessed hospital 2 dataset. Note that this data is not normalized. Each violinplot represents a feature. The title of each plot is constructed out of the first five letters of the feature name. Please check Table 4 for the full feature names.

Appendix 3

Table 16: Parameter Ids of features

Feature	**Parameter ID hospital 1**	**Parameter ID hospital 2**
creatinine	578	220615 1525
WBC	629	1542, 220546
Temperature	4	223762, 676
BSL	601	1529
HR	5	211, 220045
RR	19	220210, 618
HT	601	813, 22051
Arterial PH	650	50820
NA+	643	1536
Urea	665	1162, 225624
Bilirubin	564	50885
Albumin	527	50862
Mechanical ventilation	xxx	467, 468
Urine output	105	40055
Fio2	1063	190, 223835
MAP	-	52, 220052, 6702
PCO2	648	220235
PO2	649	220224
GCS eye	xxx	220739, 184
GCS motor	xxx	220739, 454
GCS verbal	xxx	223900, 723
DBP	11	-
SBP	10	-