Jonne Kurokallio & Vesa Vahermaa

Early Prediction of Sepsis from Clinical Data Using Generative Longitudinal Modelling

Abstract

Sepsis is one of the leading causes of death in intensive care unit (ICU) patients. Being able to react quickly to sepsis onset or successfully predict sepsis could significantly lower the mortality rates of ICU patients. In this work, we examine the viability of using various generative machine learning methods to identify time points when sepsis is imminent or has recently started. A discriminative recurrent neural network was also trained for comparison. The models use the vital signs, laboratory values, and demographic information of ICU patients for classification. Our results show that while these models can be used to predict and identify sepsis at above-chance levels, they do not reach or exceed the results obtained by other similar models in previously published literature.

Abstract **1**

Introduction **2**

Materials and Methods **3**

2.1. Data 3

2.2. Utility function and scoring method 4

2.3. Trained Models 5

2.3.1 Generative models 5

2.3.2 Long Short Term Memory (LSTM) model 6

2.3.3 Classification model 7

Results **8**

Discussion & Conclusions **9**

References **11**

Appendices **12**

Division of labor **17**

1. Introduction

Sepsis is among the leading causes of death in patients admitted to intensive care. While the mortality rate for patients who undergo sepsis remains high, they have declined significantly over time (Stevenson, Rubenstein, Radin, Wiener & Walkey, 2014). Being able to administer care quickly after sepsis occurs is vital, as a shorter reaction time to administering sepsis care is associated with lower mortality (Seymour et al., 2017). For this reason, novel methods that aim to predict sepsis before it occurs may eventually give doctors an invaluable tool for saving patients at a higher rate.

Recent research of machine learning in the context of sepsis provides some hope: for example *InSight*, a neural net classifier using multivariate combinations of easily obtained patient data was able to identify sepsis at a higher rate (0.88 AUROC at time of sepsis onset) than other clinical prediction tools such as Sequential Organ Failure Assessment (SOFA) and Systemic Inflammatory Response Syndrome (SIRS) prior to sepsis onset (Desautels et al., 2016). A more recent mixed-ward study validates *InSight's* performance as outperforming existing sepsis scoring systems in identifying and predicting sepsis, severe sepsis, and septic shock (Mao et al., 2018).

Other more recent approaches have also achieved similarly promising results: a model using a modified regularized Weibull-Cox analysis achieved a similarly high rate of sepsis identification (0.85 AUROC), this time 4 hours before sepsis onset (Nemati et al., 2018).

Consensus is yet to be reached on which features and models yield best possible prediction results. The recent PhysioNet Computing in Cardiology Challenge 2019 invited researchers to design and implement open-source algorithms that identify and predict sepsis based on vital signs and other available clinical data (Reyna et al., 2020). This report and the accompanying models are based on the PhysioNet challenge.

2. Materials and Methods

2.1. Data

The data set for the competition was collected from ICU patients in multiple hospital systems, but for this project only the A and B datasets were available. The data consists of three parts: Vital signs (8 features), laboratory values (26 features) and demographics (7 features). There are 40336 patients in total and the previously mentioned values are taken at an hourly basis. There also exist vast differences between patients' time in the ICU care. In other words, some patients have hundreds of rows of data, while some may have only ten.

Undoubtedly the largest difficulty with this data set is the huge number of missing values, denoted as NaN. As can be seen from Appendix H, 27 features are missing more than 80% of their values, some even more than 95%. Missing values greatly affect the prediction and training process, and as they are so frequent in some features, it is hard to simulate the data distributions in those features. The presented generative models can handle the missing data as they are and just ignore them during prediction. However, for the discriminative model, features with more than 80% of values missing were excluded.

Another attribute of the dataset worth paying attention to was the distribution of sepsis labels for each patient. The proportion of patients that eventually will get sepsis was much less than half, only 7.27% of the dataset. The data was divided into training and testing sets not randomly, but ensuring that 20% of sepsis patients were considered in the test set. This had a minor increase in the utility score in comparison to a random train/test distribution of the data. The initial sepsis label was also separated into two different covariates when using generative models: time to/from sepsis and sepsis label. The reason for this was to be able to base the classification on the difference in likelihoods of two hypotheses (1. the patient will not get sepsis, 2. the patient will get sepsis at time T). The time to/from sepsis holds the values of negative hours before sepsis, and after sepsis, the positive hours from the first signs of sepsis. The sepsis label is a binary label and tells if the patient is at some point going to suffer from sepsis.

When inspecting the data and contrasting the distributions of various values of individuals with identified sepsis and those with no identified sepsis, clear differences could be identified. Some of the largest differences can be seen in the heart rate, body temperature, hemoglobin, platelet, and creatinine levels, as seen in appendices A to E. These graphs include an equal number of individual measurement points taken from rows with SepsisLabel = 0 and SepsisLabel = 1. The differences are less pronounced when comparing pre-sepsis measurement points (up to 12 hours before sepsis) to other measurement points with no sepsis.

2.2. Utility function and scoring method

The scoring method for every algorithm attending the competition was determined beforehand by the organizers. A utility function was given to participants so they could evaluate the binary classification performance of their models. The utility function takes into account both cases of positive true/false and negative true/false predictions and rewards or penalizes depending on the time of sepsis prediction. The utility score is calculated for every patient separately, summed together and then normalized using the summed inaction utility and summed perfect prediction utility. This means that the score is between 0 and 1, where a score of 0 means the predictions are not useful at all. The scoring can be seen in Fig. 1.

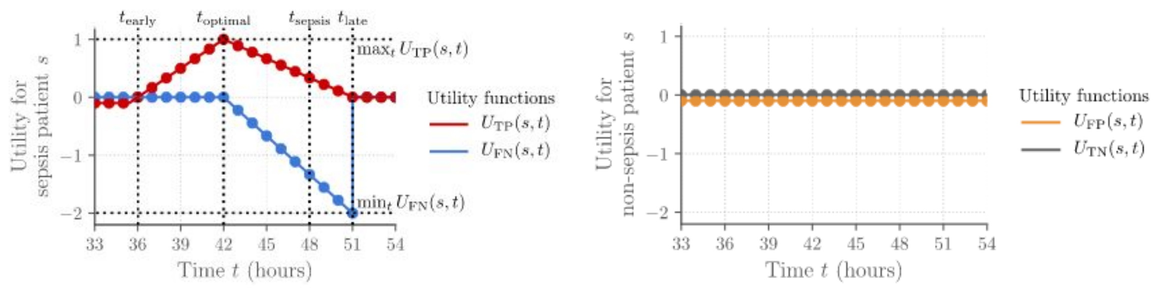


Figure 1. The utility score based on the binary classification and time of prediction

The utility function puts the weight heavily on the true positive and false negative sepsis prediction, as those predictions are the most important when it comes to the survival of the patient. For early prediction of the sepsis the score is close to zero but increases linearly when time is getting closer to the optimal time. Furthermore, too early (>12 hours prior to onset) prediction of sepsis is penalized, even if the patient does end up eventually suffering from sepsis. This is potentially due to the nature of sepsis treatment, which includes antibiotics, intravenous fluids and vasopressors, among other treatment methods ("Sepsis - Diagnosis and treatment - Mayo Clinic", 2020). Administering treatment too early might have adverse effects on the patient and potentially place unnecessary burden on the staff, while successful identification within the 12 hours prior to sepsis onset does leave enough time for the medical staff to take the appropriate precautions and prepare for treatment, including moving the patient to the ICU, if this approach is used in non-ICU environments.

In the dataset, the sepsis label variable value 1 meant that the sepsis is going to take place within the next 6 hours. The time of sepsis is the first clinical suspicion of infection or when the SOFA score of the patient drops within 24-hour period. Keeping that in mind, it makes sense to give the most points to right positive prediction at that time stamp.

Late true positive predictions are not penalized because the organ damage does not take place exactly at the time of sepsis and there might still be time to save the patient. False negatives are penalized heavily after the optimal sepsis prediction time because if that false negative prediction was followed blindly, it would eventually kill the patient.

2.3. Trained Models

2.3.1 Generative models

One of the main tasks in this project was to create multiple generative models to generate a joint probability distribution for the data. Conditional probability *P(X|Y=y)* can be calculated from this joint probability, where *X* is the observable covariates and *Y* is the target. In other words, the aim is to model the features and covariates. The simplified project pipeline for the generative models is visualized in Fig. 2 below.

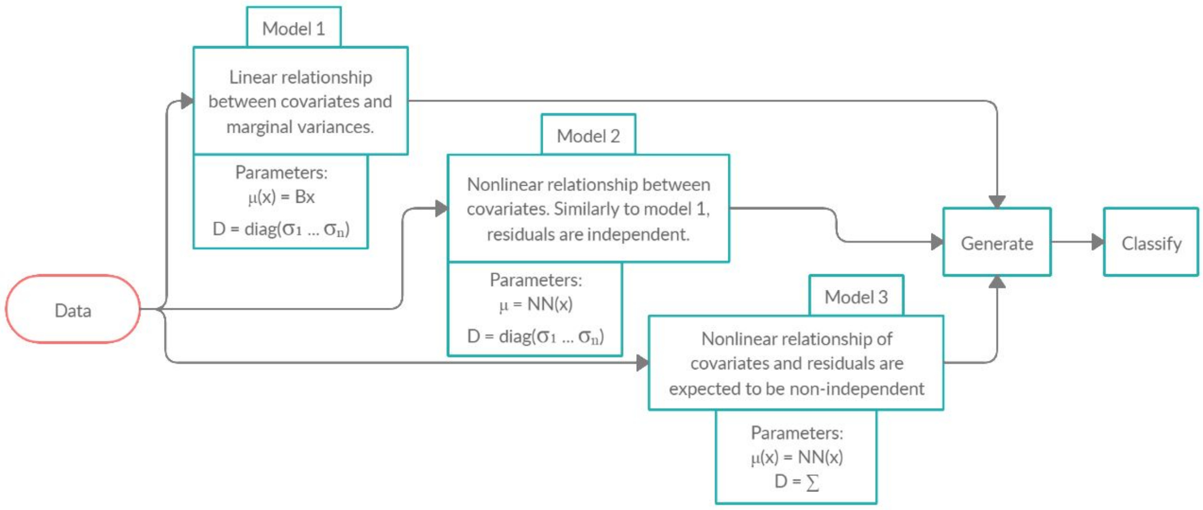

Model 1
Linear relationship between covariates and marginal variances.
Parameters: μ(x) = Bx D = diag(σ1 ... σn)
Model 2
Nonlinear relationship between covariates. Similarly to model 1, residuals are independent.
Parameters: μ = NN(x) D = diag(σ1 ... σn)
Model 3
Nonlinear relationship of covariates and residuals are expected to be non-independent
Parameters: μ(x) = NN(x) D = ∑
Data
Generate
Classify

Figure 2. Pipeline for the three generative models used with the Bayes classifier.

It is assumed that the data follows the normal distribution with mean vector μ and standard deviation matrix *D*. Our generative models follow notation where $Y = [y_1{}^T, ..., y_N{}^T]^T$ is the matrix of modelled outcomes per patient and $X = [x_1{}^T, ..., x_N{}^T]^T$ is the matrix of covariates. N is the number of rows in the data set, i.e. data points. As mentioned, the Y is assumed to be normally distributed as $y_i \sim N(\mu(x), D)$, where $\mu(x)$ is the function that maps relationship between covariates and outcome expectations in each row of the data set.

In the first model, the outcome expectations are assumed to be linearly dependent on the covariates.The mean vector is formed by $\mu(x) = Bx$, where B is the matrix of linear regression coefficients. The trainable parameters in this model are B and standard deviation matrix *D*, which is a diagonal matrix ($D = diag(\sigma^2{}_1, .., \sigma^2{}_j$)). This model assumes that the mean vector of the data distribution can be linearly drawn from the covariates, which is not the case in many real life scenarios. For example, this model is unable to detect whether old and young people are more presented in sepsis patients than middle aged people. In such a situation, the relationship should resemble more of a parabola for example.

The second model uses a normal multilayer perceptron (MLP) to apply non-linear relationship between covariates and outcome expectations ($\mu(x) = f_\theta(x)$, where f is a fully connected MLP and θ are the trainable parameters). Multiple numbers of hidden layers and layer sizes were tested in the MLP. The initial structure was two hidden layers with 14 and 28 nodes respectively, followed by ReLU activation function. The non-linearity enables the modelled outcome distributions to fall on a curve instead of a straight line, imitating real life data better than the previous model. The covariance matrix is assumed to be diagonal in this model as well.

In the third model, the standard deviation matrix is no longer considered to be diagonal. The noise is considered non-independent which means that the covariance matrix is formed as: $\Sigma_{ij} = \text{cov}(Y_i, Y_j)$. Similarly to the previous model, the MLP allows the generated mean vector to have a curvy quality ($\mu(x) = f_\theta(x)$). The residuals, however, do not have equal variance and will in this case affect each other. The covariance matrix needs to be positive definite, which means that the parameters have to be mapped to unconstrained space for the training. This mapping is done using Cholesky's decomposition $\Sigma = CC^T$, where C is unique lower triangular matrix with strictly positive diagonal values (Mohensen et al., 2011). The unconstrained parameters are fed to softplus layer and then resized to lower triangular matrix using Tensorflow's FillTriangular- function.

The math for each model is almost identical with minor changes. The probability distribution function for the normal distribution $N(\mu, \sigma^2)$ is

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} * e^{\frac{-(x-\mu)^2}{2\sigma^2}},$$

from which we can create the log-likelihood function to estimate the parameters in the following way:

$$ln\, L(\mu, \sigma^2|x_i) = \sum_{i=1}^{n} ln\, f(x|\mu, \sigma^2) = -\frac{n}{2} ln\,(2\pi) - \frac{n}{2} ln\,\sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu)^2 .$$

The log-likelihood function can be written in matrix notation for our models (select $\mu(x)$ and *D* according to the model) as:

$$ln\, L(\mu(X), D|X) = \sum_{i=1}^{n} ln\, f(X|\mu, D) = -\frac{n}{2} ln\,(2\pi) - \frac{n}{2} ln\, D - \frac{(Y-\mu(X))^T\,(Y-\mu(X))}{2D}$$

This log-likelihood is used as the loss function in the training loop.

2.3.2 Long Short Term Memory (LSTM) model

Additionally to the generative approaches, a discriminative model was also trained to the given data. The purpose was to get a deeper comparison to the generative models and also to see how a discriminative model can handle multivariate longitudinal data. After considering different types of neural networks, a recurrent neural network long-short term memory network (LSTM) was chosen, as it has proven good results in predictions involving time steps (Hochreiter et al., 1997). Where a normal fully connected layer in neural network is given a single time step as input, the LSTM layer is given a window of time steps from which it is able to extract more information in order to predict the right label. In our implementation, the data is first stripped to include only 12 features plus ID and sepsis label.

Then the data is imputed using the iterative imputer from the Scikit Learn library. This imputing method is better than deleting rows or adding mean values. The imputer models the missing features as a function of other features and then imputes the value. Most of the healthy patients that had most values missing were also excluded from the data. The remaining dataset (33089 patients) was divided into training and testing sets in the manner described in section 2.1.

The trained LSTM network consisted of one or two LSTM layers and one fully connected layer with single output and sigmoid activation function. The data was resized using a helper function to matrix [(time steps of patient-window size), window size, n_features] and it was given as input to the first LSTM layer consisting 128 units.

Similarly to gated recurrent unit, the LSTM unit has internal gates that control the data processing as can be seen in Appendix F (Yuan et al., 2019). The data is divided into two categories: important data and data that can be discarded. In every LSTM unit (cell) information is added to, or removed from, the cell state c(t) according to the gate outputs. The forget gate tells what information is relevant. It gets previous hidden state and current data input which are combined and input to sigmoid function. The input gate updates the cell state based on the combination of the hidden state and current data input. The output gate takes in the calculated new cell state and multiplies it with the combination of previous hidden state and current data input. The new produced hidden state is finally after many units used for the predictions. The structure of our final LSTM network can be seen in Fig. 3. It produces output vector size [timesteps of patient - window size, 1]. The first prediction is made after the first time window, which is why the prediction vector is smaller than the label vector. In our implementation the beginning of the prediction vector is padded with the first boolean value so that the vector lengths match.

2.3.3 Classification model

The generative models are unable to form any predictions of their own as they only portray the data distributions. A additional classifier was presented in the code template, which was used with all three generative models. The classifier evaluates simultaneously two hypotheses made according to the data distributions: 1. the patient will not suffer from sepsis at all and 2. the sepsis will happen at time point T. A probability matrix was constructed according to these two matrices by calculating the probability of either hypothesis being right. Finally, a hypothesis analysis is constructed using Bayesian factor as threshold and sepsis label 1 is assigned to the exact hour where the 2nd hypothesis is more strongly supported. The result vector is then input to the given utility function with the correct labels to receive the prediction score for that patient.

3. Results

When transforming the likelihoods into binary predictions in the classifier, we experimented with various different time windows and thresholds to investigate what the effects of changes to these parameters are. While changing these parameters did not have noticeable effect on the utility scores, the utility scores were generally higher with smaller thresholds and shorter time windows.

The exception to this is the third model, which seemed to benefit from a longer time window combined with a smaller threshold. However, the differences between the utility scores within the same model remain relatively small and consistent, as seen in Appendix G and the summary in Table 1 below. All of the generative models listed in Appendix G were trained with identical parameters (batch size 10, 100 epochs). Performance was tested with prior probabilities of 0.1 and 0.0726 in the Bayes classifier, with 0.0726 corresponding to the ratio of sepsis patients in the dataset. As seen in Appendix G, the prior probability of 0.0726 consistently got higher utility scores, and the results in Table 1 below were achieved with that prior probability.

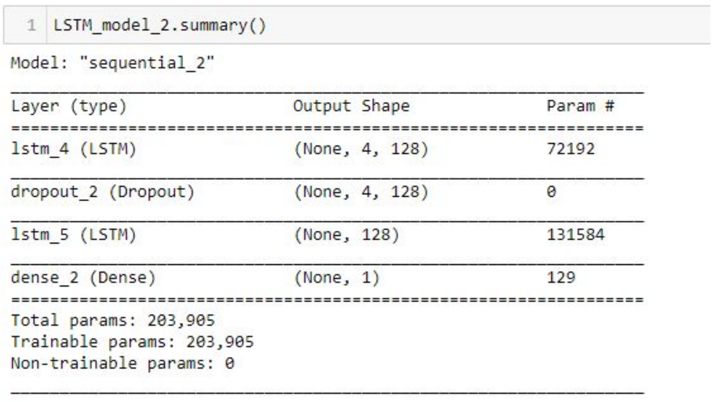
All the models listed in Appendix G were trained independently of each other, but each model was only trained once for the final results. This means that the results below are not fully reliable, as they are not averages of multiple training runs of the generative model, but of one training run of the generative model (for each model), combined with multiple (112) runs of the classifier for each model. Getting averages of multiple training runs for all the parameter combinations listed in Appendix G was impractical in the scope of this course, but it should be noted that from a conceptual perspective, Model 2 would be expected to outperform Model 1, as the relationships between the variables in vital sign data are not expected to be linear. This is not evident in the results, as we did not want to intentionally retrain the second generative model multiple times just to reach a specific conclusion, but rather trained each model once and used the Bayes classifier with the different parameter combinations and recorded the utility scores. In future research, it is recommended to train the models multiple times in order to determine the average; this is what was also done for the poster presentation of this project. However, the results in the poster presentation did not involve the 112 combinations of Bayes classifier parameters for each model.

	Model 1	Model 2	Model 3
Min utility	0.041	0.013	0.075
Mean utility	0.048	0.022	0.082
Max utility	0.051	0.028	0.086

Table 1. Utility scores with the three generative models, when tweaking parameters of the Bayes classifier used after the generative model. Prior probability 0.0726.

We also experimented with different batch sizes and epoch numbers when training the initial generative model. The best performing model, Model 3, was trained with 100, 500, and 5000 epochs (batch size 10). This did not significantly impact the results, and the utility score remained below 0.1. Due to the increase in training time, parameter tweaking of batch size and epochs was limited compared to the time window and threshold values in Appendix G. However, we were able to reach a utility score of up to 0.093 with a batch size of 30 and 100 epochs, suggesting that a larger batch size could result in better results. Still, a batch size of 50 brought the results back down to 0.055, suggesting that gaining significant benefits from increasing either the batch size or number of epochs would not necessarily yield a better model, and the primary bottleneck is likely the Bayes classifier. However, more extensive batch size and epoch number testing would be required to verify this hypothesis.

The results received from the LSTM-pipeline were a lot higher in terms of the utility score than with the generative models. The LSTM network with structure presented in fig. 3 was able to reach to 0.26 on average of five training cycles.



```
LSTM_model_2.summary()
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 4, 128)            72192
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 128)            0
_________________________________________________________________
lstm_5 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 203,905
Trainable params: 203,905
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. LSTM- network

A 4 hour timing window was used in the training of the LSTM model. Initially when the model was trained with full training data, it turned out to be heavily biased on 0 label, as it is significantly more frequent in the data set, and produced 0.000 utility score. To tackle this, label weights were introduced: 0.1 to label 0 and 0.9 to label 1. With these label weights, the LSTM model performed better, reaching a utility score of more than 0.263.

4. Discussion & Conclusions

While our models failed to outperform the PhysioNet challenge top scorers, they performed clearly above chance levels, supporting the hypothesis that machine learning methods can successfully use patients' vital signs in order to identify imminent or recent sepsis. The generative models 1-3, combined with the Bayes classifier, had a low utility score of less than 0.1, which can be attributed to the relatively simple implementation of converting the likelihood into binary predictions.

One of the most significant difficulties in using the data was the lack of laboratory measurements compared to the abundance of vital sign data. It is understandable that laboratory test values are less common than vital sign values, as collecting laboratory test values is costly, can cause patient discomfort and risk potential infection; furthermore, more frequent testing may not even yield significantly better patient outcomes (Ezzie, Aberegg & O'Brien, 2007; Kotecha, Shapiro, Cardasis & Narayanswami, 2017).

As can be seen in Table 1, model 3 outperforms both models 1 and 2. The non-linearity in the between features and covariates in addition to the non-independent residuals portray the used data most accurately.

Although the LSTM model gets good utility scores, it does not mean that it is a good model. For example in one run the LSTM model reached 0.88 binary prediction accuracy with 2372 true positives and 30899 false positives. This highlights the downside of the proposed utility scoring: it almost completely ignores false positives and true negatives, giving them close to zero score. In general, it is better to predict one than zero according to the scoring. If used in a real life scenario, our LSTM model would possibly save some lives (good utility), but also cause the personnel a lot extra work due to the false predictions. The amount of false positives could be tackled using a custom loss function which we were unable to do due to lack of skills with Tensorflow and Keras. The custom loss function could be built on calculating the recall and specificity of the predictions and assign weights so that false negatives or false positives are penalized.

There are many issues that could be done better with the LSTM model; for example, the padding of the first values makes the predictions worse with large window sizes (copying the first boolean value). Therefore the padding prevents utilizing the LSTM long memory property properly. The LSTM layers can be also used to produce a probability matrix similar to those produced with models 1-3. Using this quality could increase the model's usefulness.

A recent meta-analysis established that machine learning approaches can yield better performances than existing sepsis scoring systems in predicting sepsis (Islam et al., 2019). This strongly suggests that using a different classifier, and potentially fine-tuning the generative models, could also potentially result in a model that clearly outperforms existing scoring systems. However, more research is required before reaching a situation where a well-trained model can reliably be used in a clinical environment, especially when taking into account practical clinical factors such as questions of responsibility (e.g., for failed predictions) and other secondary effects of false positives and negatives.

5. References

Desautels, T., Calvert, J., Hoffman, J., Jay, M., Kerem, Y., & Shieh, L. et al. (2016). Prediction of Sepsis in the Intensive Care Unit With Minimal Electronic Health Record Data: A Machine Learning Approach. *JMIR Medical Informatics*, *4*(3), e28. doi: 10.2196/medinform.5909

Ezzie, M., Aberegg, S., & O'Brien, J. (2007). Laboratory Testing in the Intensive Care Unit. *Critical Care Clinics*, *23*(3), 435-465. doi: 10.1016/j.ccc.2007.07.005

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780. doi: 10.1162/neco.1997.9.8.1735

Islam, M. M., Nasrin, T., Walther, B. A., Wu, C. C., Yang, H. C., & Li, Y. C. (2019). Prediction of sepsis patients using machine learning approach: a meta-analysis. Computer methods and programs in biomedicine, 170, 1-9.

Kotecha, N., Shapiro, J., Cardasis, J., & Narayanswami, G. (2017). Reducing Unnecessary Laboratory Testing in the Medical ICU. *The American Journal Of Medicine*, *130*(6), 648-651. doi: 10.1016/j.amjmed.2017.02.014

Mao, Q., Jay, M., Hoffman, J., Calvert, J., Barton, C., & Shimabukuro, D. et al. (2018). Multicentre validation of a sepsis prediction algorithm using only vital sign data in the emergency department, general ward and ICU. *BMJ Open*, *8*(1), e017833. doi: 10.1136/bmjopen-2017-017833

Nemati, S., Holder, A., Razmi, F., Stanley, M., Clifford, G., & Buchman, T. (2018). An Interpretable Machine Learning Model for Accurate Prediction of Sepsis in the ICU. *Critical Care Medicine*, *46*(4), 547-553. doi: 10.1097/ccm.0000000000002936

Pourahmadi, M. (2011). Covariance Estimation: The GLM and Regularization Perspectives. *Statistical Science*, *26*(3), 369-387. doi: 10.1214/11-sts358

Reyna, M., Josef, C., Jeter, R., Shashikumar, S., Westover, M., & Nemati, S. et al. (2020). Early Prediction of Sepsis From Clinical Data. *Critical Care Medicine*, *48*(2), 210-217. doi: 10.1097/ccm.0000000000004145

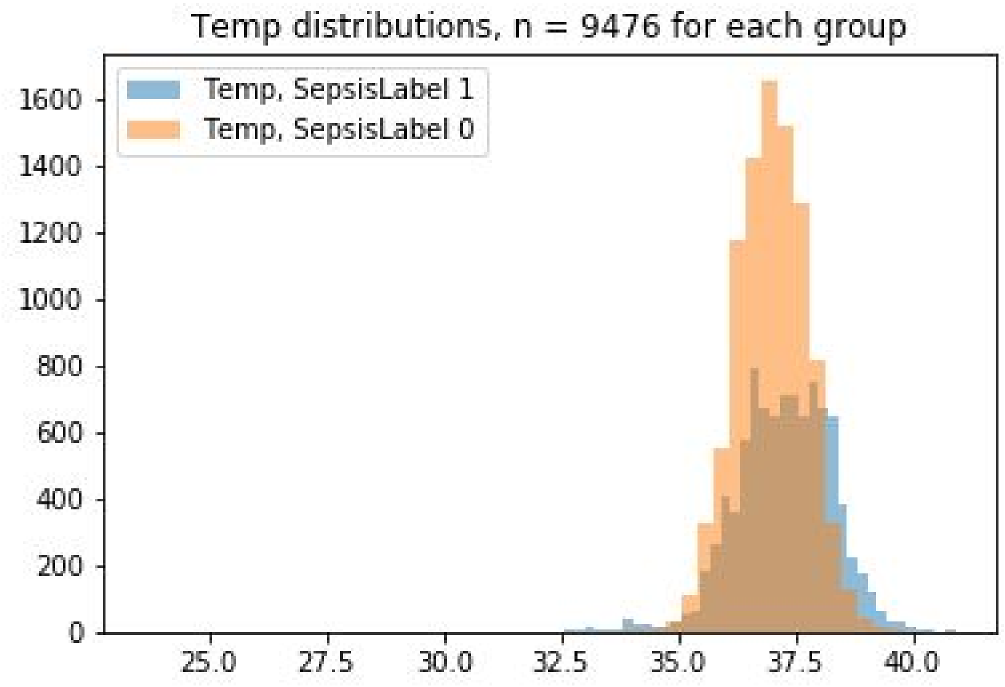
Sepsis - Diagnosis and treatment - Mayo Clinic. (2020). Retrieved 26 April 2020, from https://www.mayoclinic.org/diseases-conditions/sepsis/diagnosis-treatment/drc-20351219

Seymour, C., Gesten, F., Prescott, H., Friedrich, M., Iwashyna, T., & Phillips, G. et al. (2017). Time to Treatment and Mortality during Mandated Emergency Care for Sepsis. *New England Journal Of Medicine*, *376*(23), 2235-2244. doi: 10.1056/nejmoa1703058

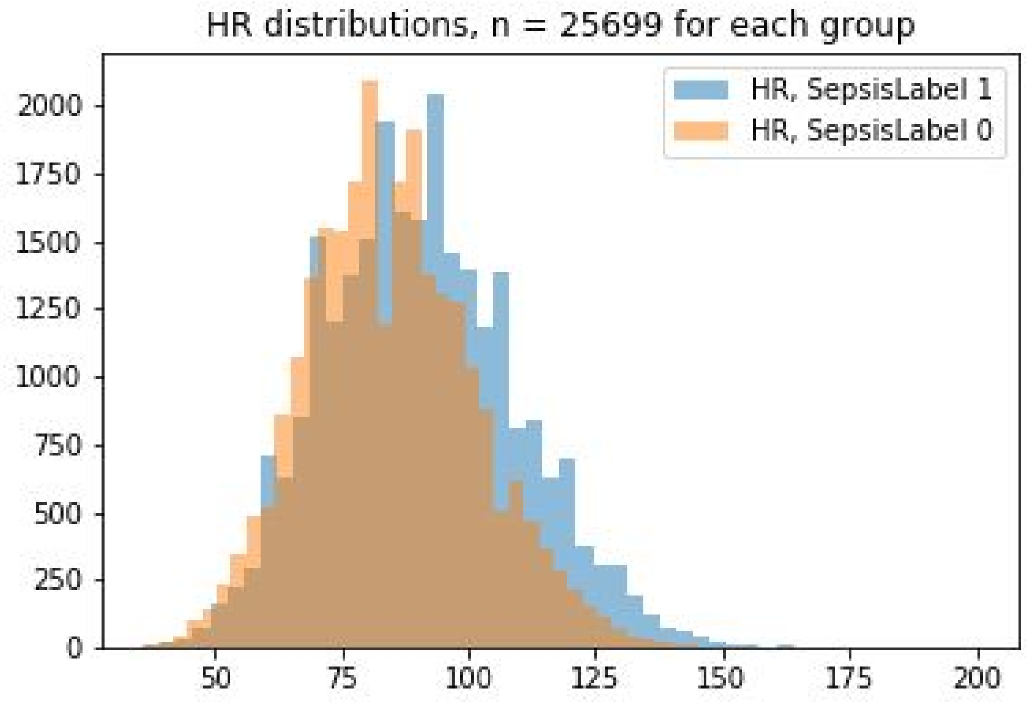
Stevenson, E., Rubenstein, A., Radin, G., Wiener, R., & Walkey, A. (2014). Two Decades of Mortality Trends Among Patients With Severe Sepsis. *Critical Care Medicine*, *42*(3), 625-631. doi: 10.1097/ccm.0000000000000026

Yuan, X., Li, L., & Wang, Y. (2020). Nonlinear Dynamic Soft Sensor Modeling With Supervised Long Short-Term Memory Network. Retrieved 20 May 2020, from https://www.researchgate.net/figure/The-structure-of-the-LSTM-unit_fig2_331421650

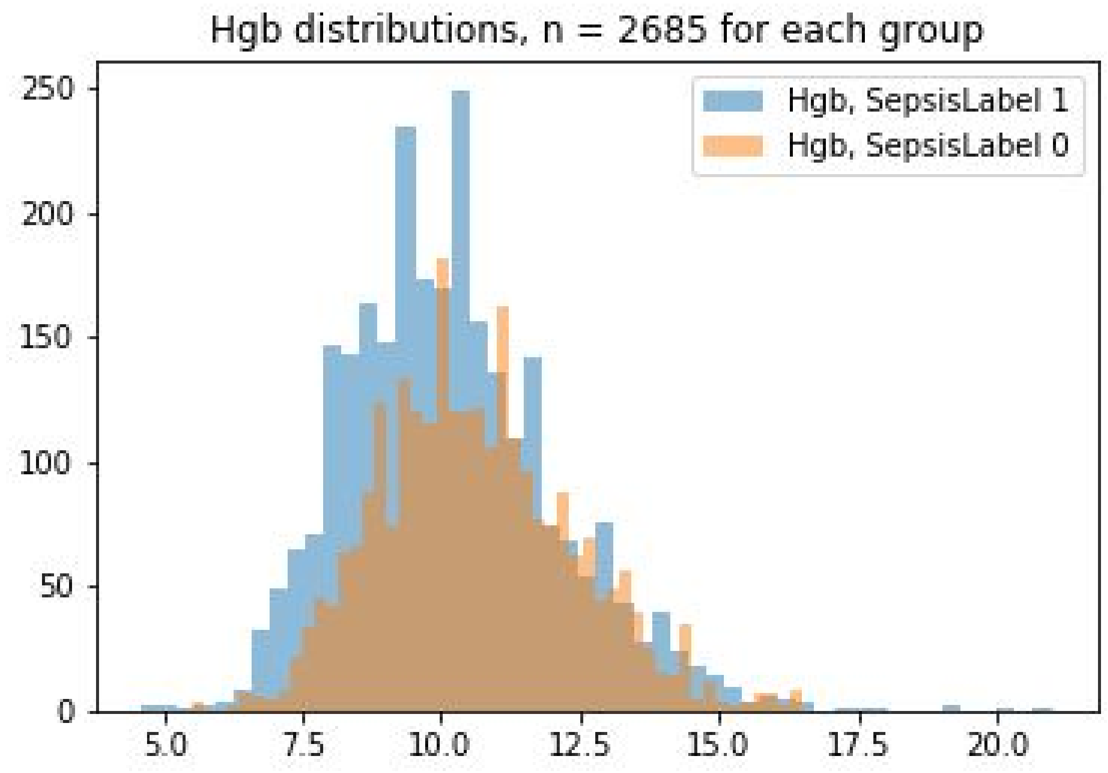
6. Appendices



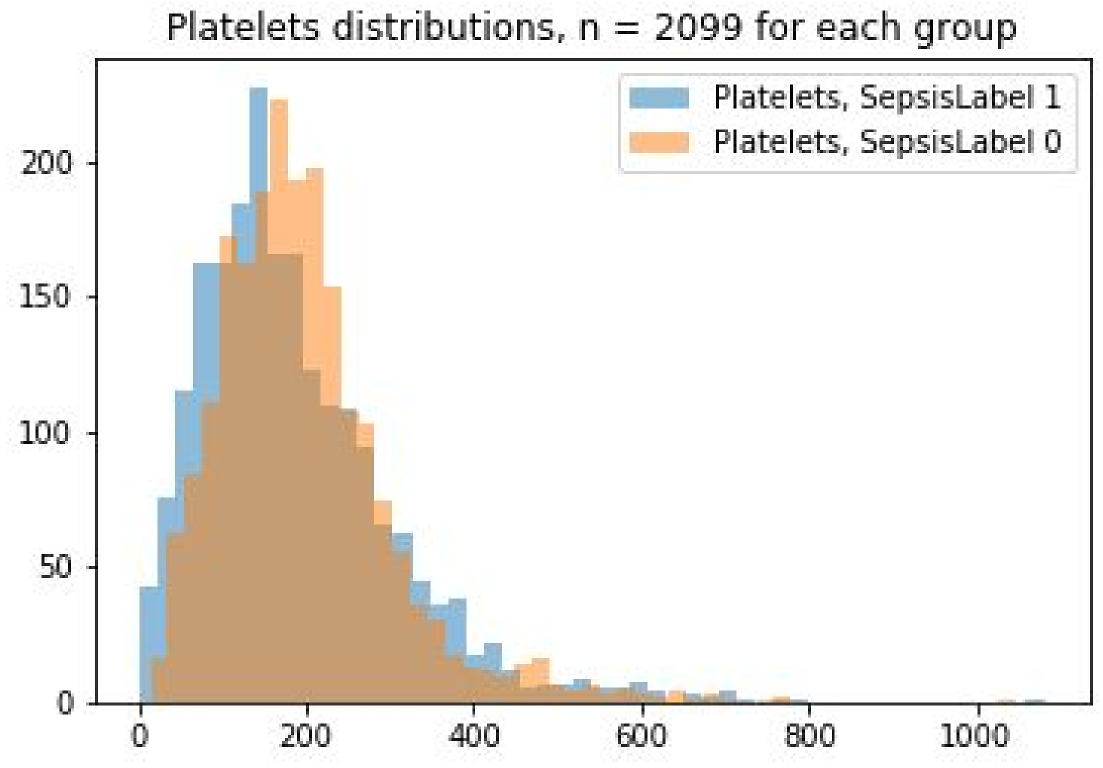
Appendix A. Body temperatures in pre/no-sepsis state and state of sepsis.



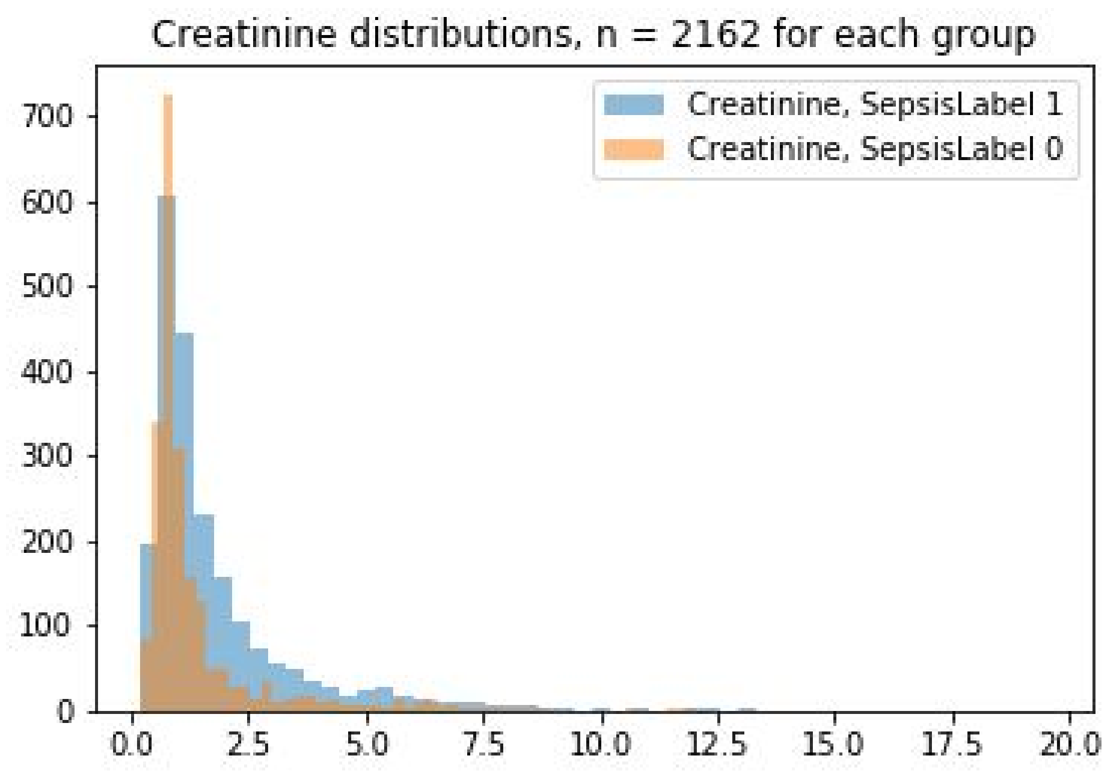
Appendix B. Heart rates in pre/no-sepsis state and state of sepsis.



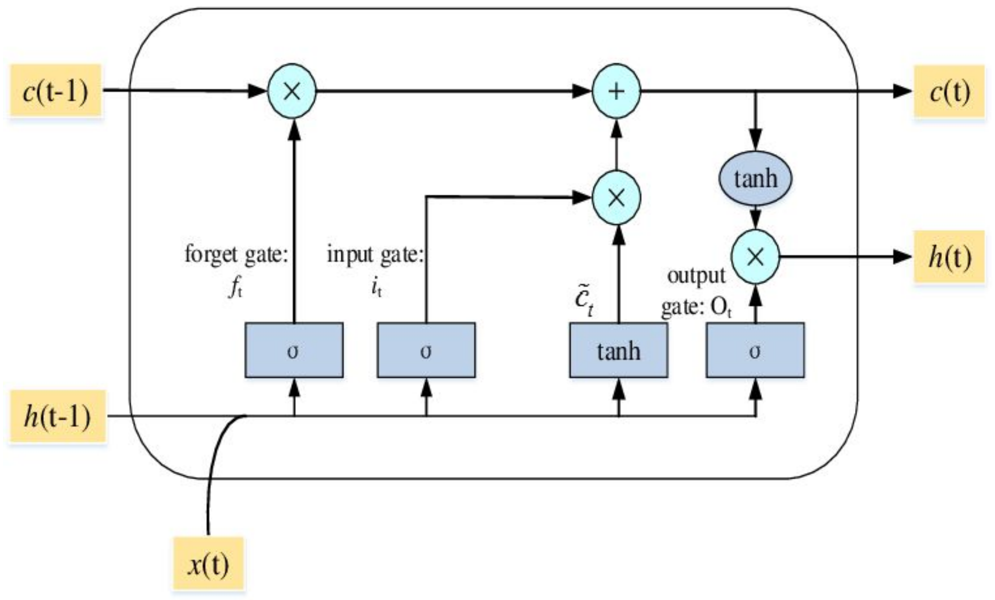
Appendix C. Hemoglobin levels in pre/no-sepsis state and state of sepsis.



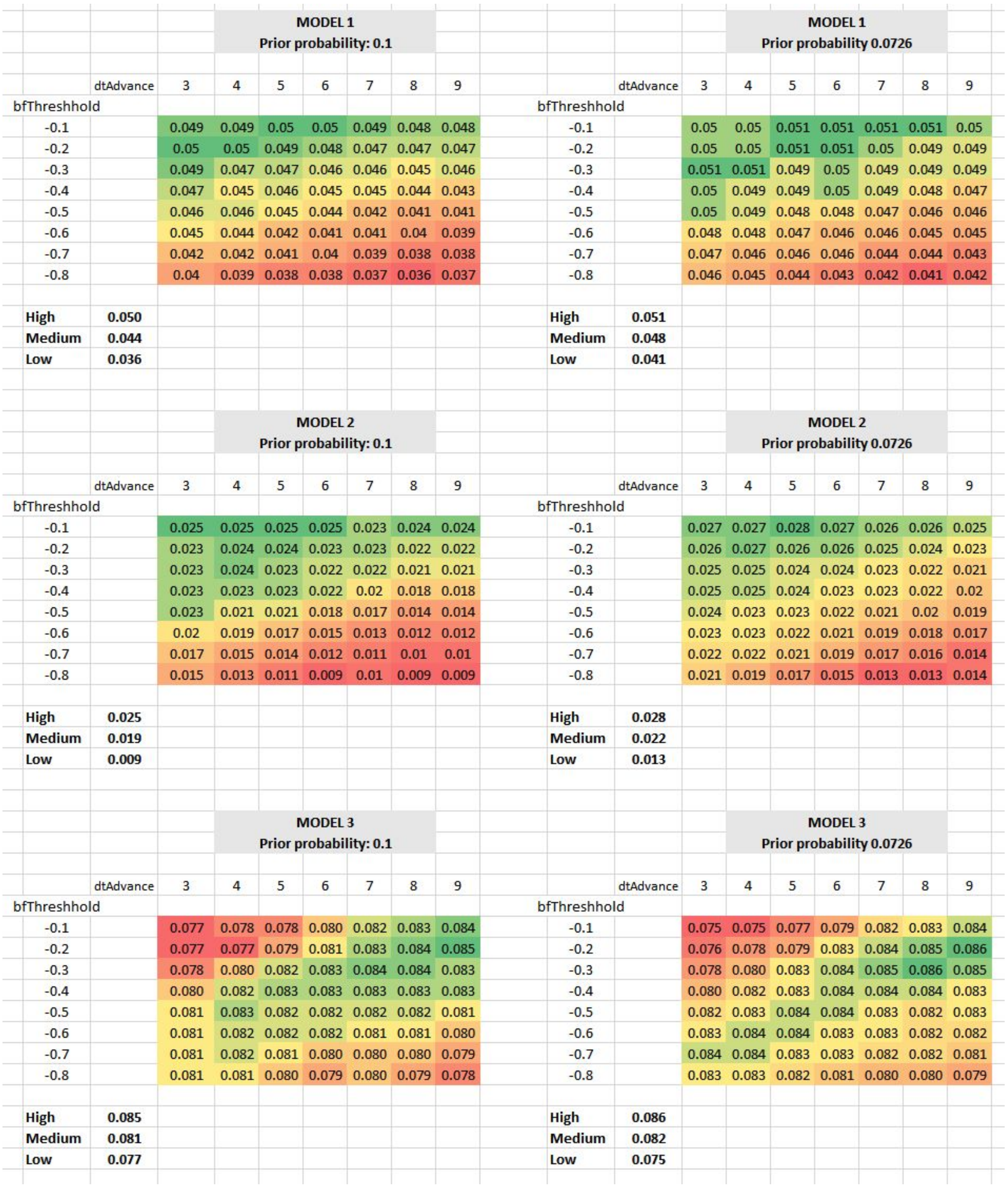
Appendix D. Platelet levels in pre/no-sepsis state and state of sepsis.



Appendix E. Creatinine levels in pre/no-sepsis state and state of sepsis.



Appendix F. The structure of the LSTM unit (Yuan et al., 2019)



MODEL 1
Prior probability: 0.1

bfThreshhold \ dtAdvance	3	4	5	6	7	8	9
-0.1	0.049	0.049	0.05	0.05	0.049	0.048	0.048
-0.2	0.05	0.05	0.049	0.048	0.047	0.047	0.047
-0.3	0.049	0.047	0.047	0.046	0.046	0.045	0.046
-0.4	0.047	0.045	0.046	0.045	0.045	0.044	0.043
-0.5	0.046	0.046	0.045	0.044	0.042	0.041	0.041
-0.6	0.045	0.044	0.042	0.041	0.041	0.04	0.039
-0.7	0.042	0.042	0.041	0.04	0.039	0.038	0.038
-0.8	0.04	0.039	0.038	0.038	0.037	0.036	0.037

High	**0.050**
Medium	**0.044**
Low	**0.036**

MODEL 1
Prior probability 0.0726

bfThreshhold \ dtAdvance	3	4	5	6	7	8	9
-0.1	0.05	0.05	0.051	0.051	0.051	0.051	0.05
-0.2	0.05	0.05	0.051	0.051	0.05	0.049	0.049
-0.3	0.051	0.051	0.049	0.05	0.049	0.049	0.049
-0.4	0.05	0.049	0.049	0.05	0.049	0.048	0.047
-0.5	0.05	0.049	0.048	0.048	0.047	0.046	0.046
-0.6	0.048	0.048	0.047	0.046	0.046	0.045	0.045
-0.7	0.047	0.046	0.046	0.046	0.044	0.044	0.043
-0.8	0.046	0.045	0.044	0.043	0.042	0.041	0.042

High	**0.051**
Medium	**0.048**
Low	**0.041**

MODEL 2
Prior probability: 0.1

bfThreshhold \ dtAdvance	3	4	5	6	7	8	9
-0.1	0.025	0.025	0.025	0.025	0.023	0.024	0.024
-0.2	0.023	0.024	0.024	0.023	0.023	0.022	0.022
-0.3	0.023	0.024	0.023	0.022	0.022	0.021	0.021
-0.4	0.023	0.023	0.023	0.022	0.02	0.018	0.018
-0.5	0.023	0.021	0.021	0.018	0.017	0.014	0.014
-0.6	0.02	0.019	0.017	0.015	0.013	0.012	0.012
-0.7	0.017	0.015	0.014	0.012	0.011	0.01	0.01
-0.8	0.015	0.013	0.011	0.009	0.01	0.009	0.009

High	**0.025**
Medium	**0.019**
Low	**0.009**

MODEL 2
Prior probability 0.0726

bfThreshhold \ dtAdvance	3	4	5	6	7	8	9
-0.1	0.027	0.027	0.028	0.027	0.026	0.026	0.025
-0.2	0.026	0.027	0.026	0.026	0.025	0.024	0.023
-0.3	0.025	0.025	0.024	0.024	0.023	0.022	0.021
-0.4	0.025	0.025	0.024	0.023	0.023	0.022	0.02
-0.5	0.024	0.023	0.023	0.022	0.021	0.02	0.019
-0.6	0.023	0.023	0.022	0.021	0.019	0.018	0.017
-0.7	0.022	0.022	0.021	0.019	0.017	0.016	0.014
-0.8	0.021	0.019	0.017	0.015	0.013	0.013	0.014

High	**0.028**
Medium	**0.022**
Low	**0.013**

MODEL 3
Prior probability: 0.1

bfThreshhold \ dtAdvance	3	4	5	6	7	8	9
-0.1	0.077	0.078	0.078	0.080	0.082	0.083	0.084
-0.2	0.077	0.077	0.079	0.081	0.083	0.084	0.085
-0.3	0.078	0.080	0.082	0.083	0.084	0.084	0.083
-0.4	0.080	0.082	0.083	0.083	0.083	0.083	0.083
-0.5	0.081	0.083	0.082	0.082	0.082	0.082	0.081
-0.6	0.081	0.082	0.082	0.082	0.081	0.081	0.080
-0.7	0.081	0.082	0.081	0.080	0.080	0.080	0.079
-0.8	0.081	0.081	0.080	0.079	0.080	0.079	0.078

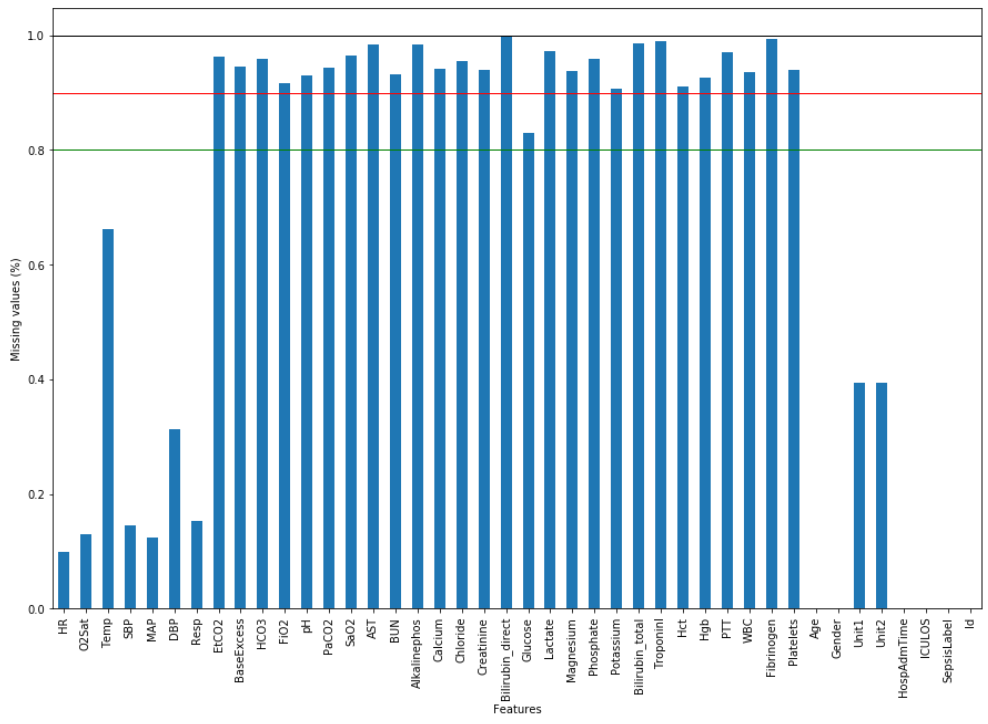
High	**0.085**
Medium	**0.081**
Low	**0.077**

MODEL 3
Prior probability 0.0726

bfThreshhold \ dtAdvance	3	4	5	6	7	8	9
-0.1	0.075	0.075	0.077	0.079	0.082	0.083	0.084
-0.2	0.076	0.078	0.079	0.083	0.084	0.085	0.086
-0.3	0.078	0.080	0.083	0.084	0.085	0.086	0.085
-0.4	0.080	0.082	0.083	0.084	0.084	0.084	0.083
-0.5	0.082	0.083	0.084	0.084	0.083	0.082	0.083
-0.6	0.083	0.084	0.084	0.083	0.083	0.082	0.082
-0.7	0.084	0.084	0.083	0.083	0.082	0.082	0.081
-0.8	0.083	0.083	0.082	0.081	0.080	0.080	0.079

High	**0.086**
Medium	**0.082**
Low	**0.075**

Appendix G. Utility scores with generative models 1-3 & Bayes classifier



Appendix H. The percentage of missing values per feature

7. Division of labor

The course work was done in a group of two, participants Jonne Kurokallio and Vesa Vahermaa. Jonne’s main responsibility was for developing the learning models (models 1-3, as well as the LSTM model) that achieved the final results, while Vesa did data analysis and visualization as well as literature review and tuning of the models 1-3 and the Bayes classifier based on changing the parameters. The poster for the poster presentation was composed by both parties, and the introduction video accompanying the poster was done by Vesa. Both participants contributed to writing of this final report, with Vesa writing all of chapter 1 and half of 4, and Jonne writing most of 2 and most of 3. All in all the workload was divided almost 50/50.