Early prediction of sepsis using LSTM networks

by

Congxing Zhu

B. S., Yangzhou University, China, 2012

M. S., Kansas State University, 2019

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2020

Approved by:

Major Professor
Dr. Doina Caragea

Copyright

Abstract

Sepsis is a severe life-threatening disease that causes millions of deaths every year. As a significant public health problem, the annual costs of hospital treatment of sepsis patients are rising. Without the treatment of antibiotics for sepsis, the risk of mortality increases with each additional hour. It is desirable to diagnose sepsis as early as possible, ideally earlier than the doctors can identify it with traditional methods. Many existing machine learning approaches show excellent results for onset sepsis prediction. However, it is important to perform early prediction of sepsis (i.e., hours before it takes place), especially in the case of intensive care unit patients. In this thesis, I train a Long Short-Term Memory (LSTM) network for the early prediction of sepsis using an existing dataset published as part of the PhysioNet/Computing in Cardiology Challenge in 2019. I show that adding a variety of predictors based on existing hourly recorded variables significantly improves the prediction results. I also show how tuning the number of epochs and observing the learning curves can lead to better outcomes for sepsis's early prediction. Finally, I show that using data standardization destroys the diversity and variety in the data and weakens the early prediction outcomes. My best model achieves a utility score of 0.421, an F-score of 0.128, and an area of 0.129 under the precision-recall curve on the publicly available data. The obtained standardized utility score is slightly higher than those reported in some published papers, which use Long Short-Term Memory networks. However, for clinical application, improved approaches would be needed to produce more reliable results.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would especially like to thank my adviser Dr. Doina Caragea. Without her help and support, this work would not be possible. I want to thank my master's committee members, Dr. Mitchell Neilsen and Dr. Torben Amtoft, for their helpful comments and suggestions during my research. I want to thank PhysioNet/Computing in Cardiology Challenge for sharing the data for this project.

Chapter 1

Introduction

Sepsis is a public health problem that is life-threatening to patients and costly to hospitals (Singer et al., 2016). The early application of optimal sepsis care can lower the risk of in-hospital mortality (Seymour et al., 2017), as the delayed antimicrobial therapy for sepsis patients results in a higher mortality rate (Frost et al., 2010). One of the major issues is that many sepsis patients are not diagnosed at admission, making the early treatment impossible. Therefore, early prediction of sepsis and the application of antimicrobial treatments promptly for sepsis patients are greatly needed.

In general, the hospital will record the patients' health condition hourly after they enter the intensive care unit (ICU). Many researchers concentrate on using this data to improve the sepsis prediction accuracy and achieve excellent outcomes using machine learning approaches (e.g., Desautels et al., 2016; Fleuren et al., 2020). However, using the time-series data to predict sepsis early is challenging, despite its significant benefits. Intuitively, recurrent neural networks (RNNs) can be used for early sepsis prediction from sequential time-series data. The reason is that RNNs are specifically designed to learn from sequential data and effectively store and use past information. But the general RNNs suffer from vanishing and exploding gradients, especially when the length of the sequence, particularly the patient's length of stay, is large. Long Short-Term Memory (LSTM) networks are recurrent neural networks that can capture both short and long term dependencies and are less vulnerable to the

vanishing gradient problem. Many researchers have shown that training and evaluating the LSTM model on the PhysioNet/Computing in Cardiology Challenge 2019 data can produce excellent results (e.g., Nejedly et al., 2019; Schellenberger et al., 2019; Wang et al., 2019; Roussel et al., 2019; He et al., 2019; Vicar et al., 2019). However, implementing different approaches may differ in the early prediction of sepsis results. For instance, some researchers implement a special data normalization approach on the variables and obtain a normalized utility score of 0.281 (Vicar et al., 2019). However, other researchers achieve a better score of 0.29 without scaling the data (Schellenberger et al., 2019). In addition, some researchers conduct the experiment without expanding the number of variables and obtain good results (e.g., Nejedly et al., 2019; Vicar et al., 2019). However, several research groups report higher normalized utility scores by adding more variables based on the original 40 variables (e.g., He et al., 2019; Schellenberger et al., 2019; Wang et al., 2019; Roussel et al., 2019). All the researchers train and evaluate their LSTM models to generate the best prediction results. Nevertheless, not all researchers describe the process of tuning the number of epochs. Additionally, the researchers use different approaches to solve the missing value issue, such as replacing them with a constant value (Roussel et al., 2019) and using forward filling (Nejedly et al., 2019).

In this thesis, I compare the differences between eight different experimental settings, including all the scenarios I mentioned above. I also tune the number of epochs and find that the number of epochs makes a big difference in the early prediction of sepsis. The workflow used in my thesis is summarized in Figure 1.1. First of all, I pre-process the original data by splitting the data and cleaning data. I then train the model and evaluate the trained model on the test data for each implementation setting. In the end, I summarize the results and compare the differences between different combinations of settings. This thesis's experimental results show that adding various predictors based on existing hourly recorded variables and tuning the number of epochs can significantly improve the prediction results. In addition, my thesis shows that using data standardization destroys diversity and variety in the data and weakens the early prediction outcomes. Schellenberger et al. (2019) and Roussel et al. (2019) who use LSTM networks for the challenge reported their best

utility score on publicly available datasets in their published papers. Their best scores are 0.408 and 0.393, respectively. My thesis's best model achieves a utility score of 0.419, which is slightly higher than their utility scores.

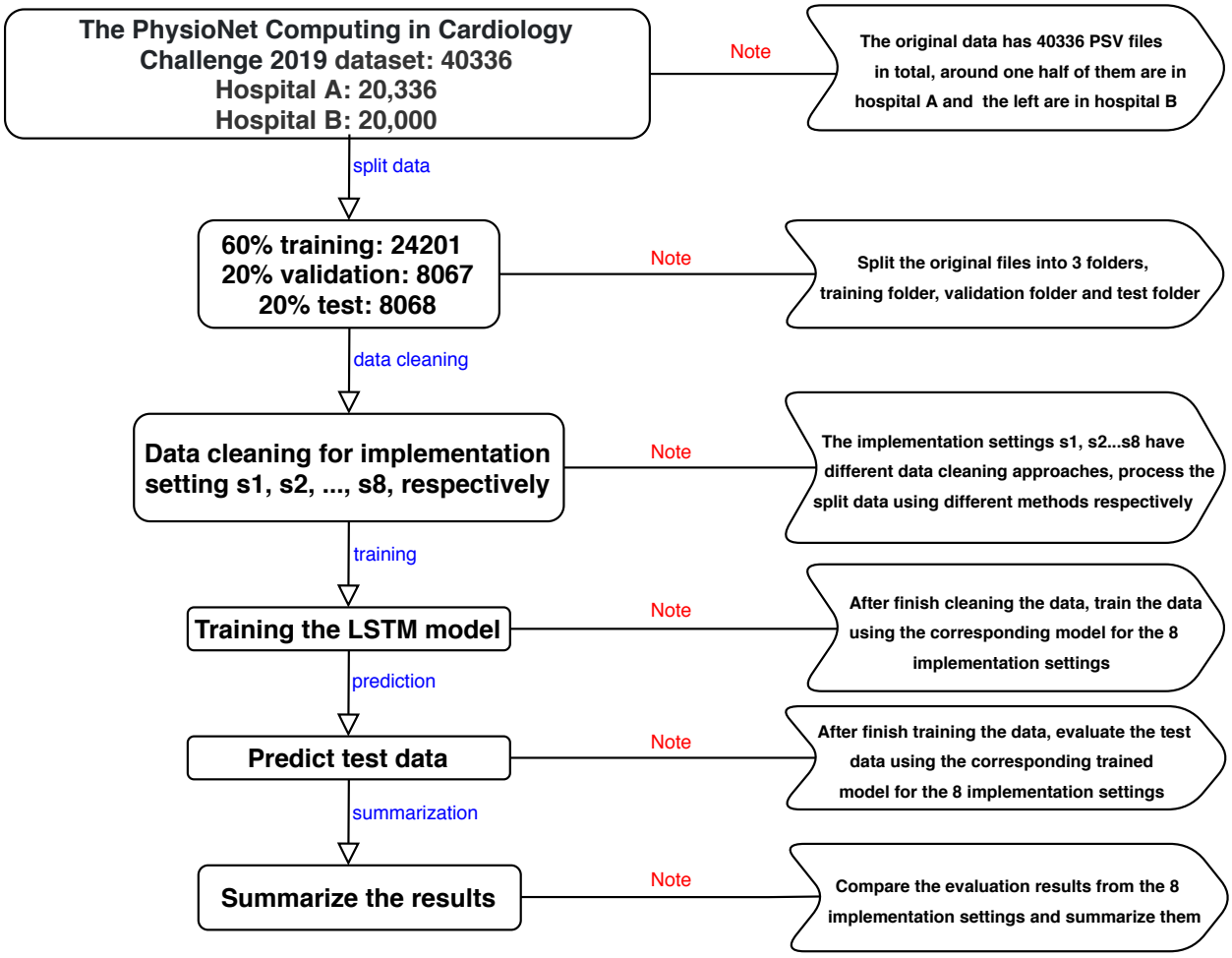


Figure 1.1: Overview of the workflow for sepsis prediction. The left part of the flowchart represents the whole process, from pre-processing the data to summarizing the results. The right side presents more details for each method.

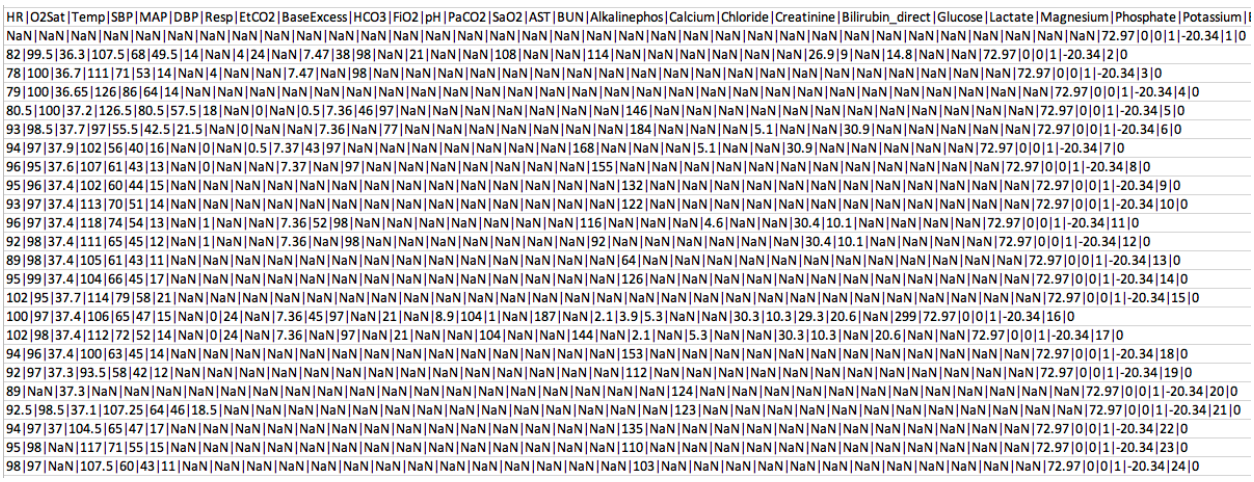
The rest of the thesis is organized as follows: Chapter 2 describes the data and pre-processing approaches; Chapter 3 reviews the LSTM networks and describes how to apply LSTM for the sepsis data that was used in the thesis; Chapter 4 presents the experimental setup; Chapter 5 illustrates the implementation of the Python code created in a Jupyter Notebook; Chapter 6 shows the results; Chapter 7 includes a discussion of the results, and finally Chapter 8 presents conclusions, and ideas for future work.

Chapter 2

Sepsis data

2.1 Data description

The PhysioNet/Computing in Cardiology Challenge 2019 data consists of 40336 PSV files, 20336 in hospital A and 20000 in hospital B (Reyna et al., 2019). Each PSV file refers to one patient's hourly recorded health condition data after entering the ICU. Figure 2.1 shows the details in one PSV file. The hourly collected data on each patient consists of 41 variables, as shown in Table 2.1. The first 40 variables include eight vital signs, such as heart rate, pulse oximetry, and temperature. The remaining 32 variable include 26 laboratory values (Measure of excess bicarbonate, Bicarbonate, Fraction of inspired oxygen, etc.), and six demographics (age, gender, ICU length-of-stay, etc.). The last variable is the sepsis label. The label is recorded hourly using 0 or 1, where 0 means no sepsis, and 1 means that sepsis is identified based on Sepsis-3 criteria (Singer et al., 2016).



HR|O2Sat|Temp|SBP|MAP|DBP|Resp|EtCO2|BaseExcess|HCO3|FiO2|pH|PaCO2|SaO2|AST|BUN|Alkalinephos|Calcium|Chloride|Creatinine|Bilirubin_direct|Glucose|Lactate|Magnesium|Phosphate|Potassium|I
NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|1|0
82|99.5|36.3|107.5|68|49.5|14|NaN|4|24|NaN|7.47|38|98|NaN|21|NaN|NaN|108|NaN|NaN|114|NaN|NaN|NaN|NaN|NaN|NaN|26.9|9|NaN|14.8|NaN|NaN|72.97|0|0|1|-20.34|2|0
78|100|36.7|111|71|53|14|NaN|4|NaN|NaN|7.47|NaN|98|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|3|0
79|100|36.65|126|86|64|14|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|4|0
80.5|100|37.2|126.5|80.5|57.5|18|NaN|0|NaN|0.5|7.36|46|97|NaN|NaN|NaN|NaN|NaN|NaN|NaN|146|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|5|0
93|98.5|37.7|97|55.5|42.5|21.5|NaN|0|NaN|NaN|7.36|NaN|77|NaN|NaN|NaN|NaN|NaN|NaN|NaN|184|NaN|NaN|NaN|5.1|NaN|NaN|30.9|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|6|0
94|97|37.9|102|56|40|16|NaN|0|NaN|0.5|7.37|43|97|NaN|NaN|NaN|NaN|NaN|NaN|NaN|168|NaN|NaN|NaN|5.1|NaN|NaN|30.9|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|7|0
96|95|37.6|107|61|43|13|NaN|0|NaN|NaN|7.37|NaN|97|NaN|NaN|NaN|NaN|NaN|NaN|NaN|155|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|8|0
95|96|37.4|102|60|44|15|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|132|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|9|0
93|97|37.4|113|70|51|14|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|122|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|10|0
96|97|37.4|118|74|54|13|NaN|1|NaN|NaN|7.36|52|98|NaN|NaN|NaN|NaN|NaN|NaN|NaN|116|NaN|NaN|NaN|4.6|NaN|NaN|30.4|10.1|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|11|0
92|98|37.4|111|65|45|12|NaN|1|NaN|NaN|7.36|NaN|98|NaN|NaN|NaN|NaN|NaN|NaN|NaN|92|NaN|NaN|NaN|NaN|NaN|NaN|30.4|10.1|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|12|0
89|98|37.4|105|61|43|11|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|64|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|13|0
95|99|37.4|104|66|45|17|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|126|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|14|0
102|95|37.7|114|79|58|21|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|15|0
100|97|37.4|106|65|47|15|NaN|0|24|NaN|7.36|45|97|NaN|21|NaN|8.9|104|1|NaN|187|NaN|2.1|3.9|5.3|NaN|NaN|30.3|10.3|29.3|20.6|NaN|299|72.97|0|0|1|-20.34|16|0
102|98|37.4|112|72|52|14|NaN|0|24|NaN|7.36|NaN|97|NaN|21|NaN|NaN|104|NaN|NaN|144|NaN|2.1|NaN|5.3|NaN|NaN|30.3|10.3|NaN|20.6|NaN|NaN|72.97|0|0|1|-20.34|17|0
94|96|37.4|100|63|45|14|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|153|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|18|0
92|97|37.3|93.5|58|42|12|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|112|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|19|0
89|NaN|37.3|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|124|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|20|0
92.5|98.5|37.1|107.25|64|46|18.5|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|123|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|21|0
94|97|37|104.5|65|47|17|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|135|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|22|0
95|98|NaN|117|71|55|15|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|110|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|23|0
98|97|NaN|107.5|60|43|11|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|103|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|72.97|0|0|1|-20.34|24|0

Figure 2.1: Example of a PSV file, which is similar to a CSV file. The PSV (pipe separated values) file uses a | "pipe" as the delimiter character, instead of a comma character in the CSV (comma-separated values) file. The first row shows the 41 variables in the sepsis dataset, while the remaining rows present the patient's hourly health record.

The data is highly imbalanced, with only 2932 out of the total of 40336 PSV files/patients having sepsis, which accounts for 7.27% of all patients. In addition, 26 variables out of 41 variable have more than 90% missing values, especially laboratory values. To enable early sepsis prediction, the sepsis label has been moved forward by six hours for all the datasets (meaning that the label is set to 1 for six hours before it is officially identified).

Table 2.1: The names of the 41 variables in the sepsis dataset, including eight vital signs, 26 laboratory values, six demographics, and one outcome. The outcome is the sepsis label, as highlighted in blue, and is the variable that we want to predict.

Vital signs:	HR, O2Sat, Temp, SBP, MAP, DBP, Resp, EtCO2
Laboratory values:	BaseExcess, HCO3, FiO2, pH, PaCO2, SaO2, AST, BUN, Alkalinephos, Calcium, Chloride, Creatinine, Bilirubin_direct, Glucose, Lactate, Magnesium, Phosphate, Potassium, Bilirubin_total, TroponinI, Hct, Hgb, PTT, WBC, Fibrinogen, Platelets
Demographics:	Age, Gender, Unit1, Unit2, HospAdmTime, ICULOS
Outcome:	SepsisLabel

2.2 Data pre-processing

For pre-processing the data in my thesis, I first combine all the PSV files from two hospitals into one folder. I then split the whole datasets into three parts: 60% training, 20% validation, and 20% test sets.

I implement eight different settings and evaluate their performances by comparing the differences in the utility score. The settings are obtained by taking combinations of the following two operations: 1) performing standardization or not; and 2) Adding predictors or not. The utility scores are calculated using the method provided by the PhysioNet Computing in Cardiology Challenge 2019 (Reyna et al., 2019). The method rewards for early prediction and punishes for the late prediction of sepsis. I conduct the data pre-processing differently, depending on the setting used. For example, the first setting uses "no standardization" and "no adding predictors", which means I do not standardize the data and use the original 40 variables without adding more predictors before fitting into the training model. If standardization is performed, I scale the original data by standardizing them between 0 to 1, except the sepsis label. In general, if the features have different ranges, we could standardize the data before fitting them into the training model. I compare the differences between standardization and non-standardization because some researchers obtain better results without standardizing the data (Schellenberger et al., 2019).

In the setting where no predictors are adding, I use the original 40 variables as the predictors in the model without adding additional predictors. I replace the missing values using forward and backward filling methods. The forward filling is conducted if there is a recorded value before the current time, while the backward filling is implemented when there is no prior record with a value, as it generally happens in the first several hours. For the setting when more predictors are added, I copy the first 34 variables and append them into the original dataset. I then use forward filling and backward filling to replace missing values for the appended variables. Meanwhile, the missing values for the original first 40 variables are replaced with 0. I also compute the difference between the current hour record and the previous hour for the appended 34 variables and add them to the dataset. The

reason for adding the difference is that the current values may change much compared to the last hour's values when the doctor identifies sepsis. I assume the hourly difference can capture the signal of diagnosing sepsis. Many researchers have shown that expanding the predictors using the original 40 variables improves the prediction results (e.g., Wang et al., 2019; Roussel et al., 2019; Schellenberger et al., 2019). The reason that I ignore copying the last six groups 'age', 'gender', 'unit1', 'unit2', 'HospAdmTime', and 'ICULOS' is that those groups have consistent values. Adding them will generate repeated values, while computing the differences will generate many 0s. Thus, I assumed that they cannot improve model performance. Therefore, adding more predictors will result in 108 variables in total, the 40 original predictors plus the newly generated 68 predictors.

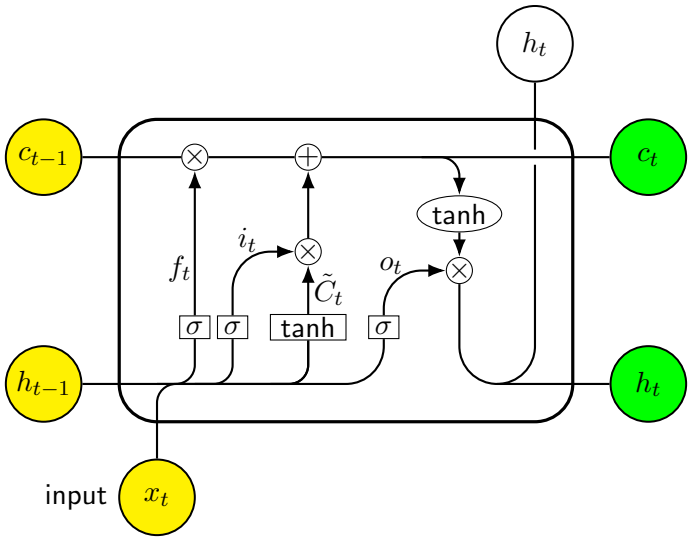
Chapter 3

Methods: Long Short-Term Memory

Introduced by Hochreiter and Schmidhuber (1997), Long Short-Term Memory networks are a special type of recurrent neural network capable of learning long-term dependencies in sequential data. LSTMs have been widely used on various of problems, such as nature language processing (Sutskever et al., 2014), forecasting stock prices (Qiu et al., 2020), and text classification (Luan and Lin, 2019).

As shown in Figure 3.1, an LSTM network consists of three gates: input gate i_t, forget gate f_t, and output gate o_t, and a cell unit to update and store the previous information for the sequential data. The forget gate f_t decides how much information to keep from the cell state C_{t-1}. The operator σ represents a sigmoid function, which produces a number between 0 and 1 for each number in the cell state C_{t-1} by using the information from h_{t-1} and x_t. Before merging into the cell unit, the input gate i_t combines with $\tilde{C}_t$ computed by a tanh function from h_{t-1} and x_t. After applying the tanh function for the current unit cell, we combine it with the output gate o_t, which will generate h_t. Finally, after executing the multiplication operator for the unite cell C_{t-1} and forget gate, the plus operator is used on the current unit cell and input gate, and the output is the new cell state C_t. The hidden state h_t and the cell state C_t will be used for next timestep.

In order to fit the sepsis sequence data of a patient into the LSTM model, for each PSV file, I generate an "instance" for each timestep t in that file. The instance corresponding



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

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

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

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

Figure 3.1: LSTM network: The left side shows an LSTM network architecture with the corresponding computation flow, while the right side formulas show the complex operations inside the left block. W_f, W_i, W_o and $b_f.b_i, b_o$ are the training parameters in the LSTM network. The signs $\otimes$, $\oplus$, $*$, and $\cdot$ represent multiplication operator, plus operator, dot product, and matrix multiplication, respectively.

to timestep t is a sequence consisting of all the records corresponding to timesteps prior to the current timestep t, together with the record at the current timestep t. The label of the instance is the sepsis label corresponding to the current timestep t. For example, if the PSV file of a patient has 30 timesteps, the first instance will include the first timestep record. The second instance consists of the first and second timesteps' records. The thirtieth instance contains all the hourly records in the file. For each PSV file, the data will then become a list, including many arrays. Each array corresponds the all the records at and before this current time. The left side of Figure 3.2 shows the original data without sepsis, and the right side displays what the data look like after the processing. Similar to Figure 3.2, Figure 3.3 presents the data with sepsis. The sepsis label has been moved six hours earlier in the PhysioNet Computing in Cardiology Challenge 2019 datasets. Otherwise, we need to move all the sepsis labels by ourselves. For example, all the sepsis labels on the right side of Figure 3.3 should be set to 1. The reason is the sepsis was identified at t7, and moving six hours earlier means t1 should be 1, too.

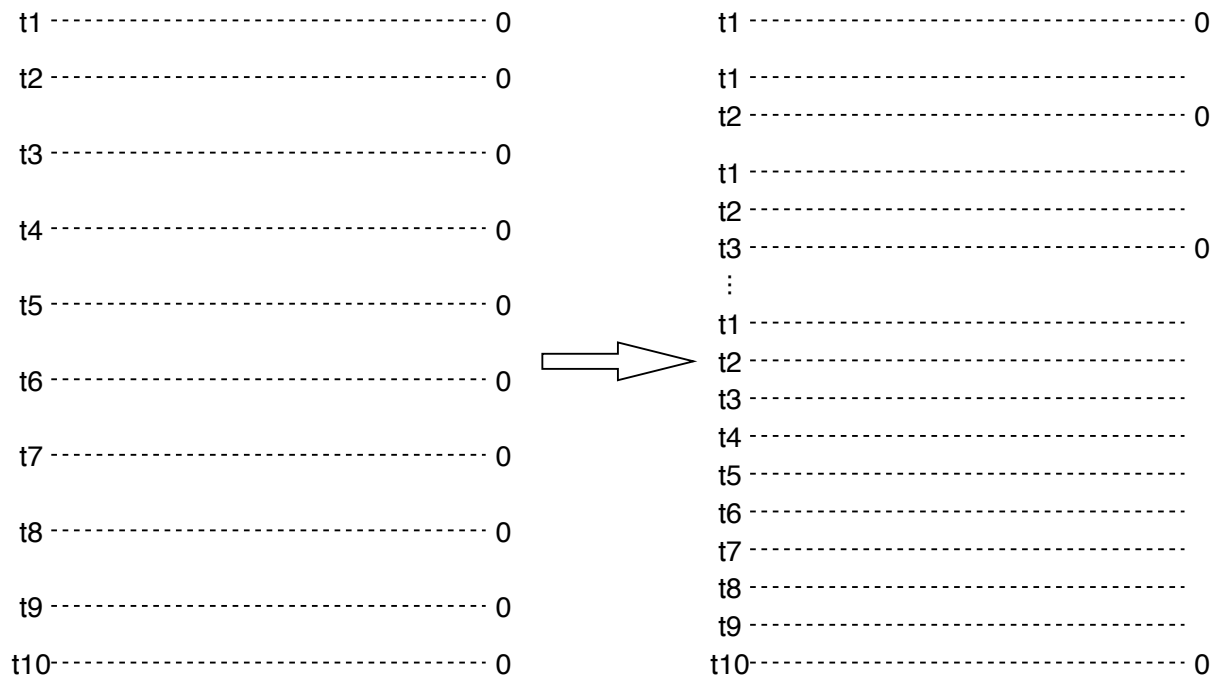


Figure 3.2: The left side shows the original data without sepsis, and the right side displays what the data look like after the processing.

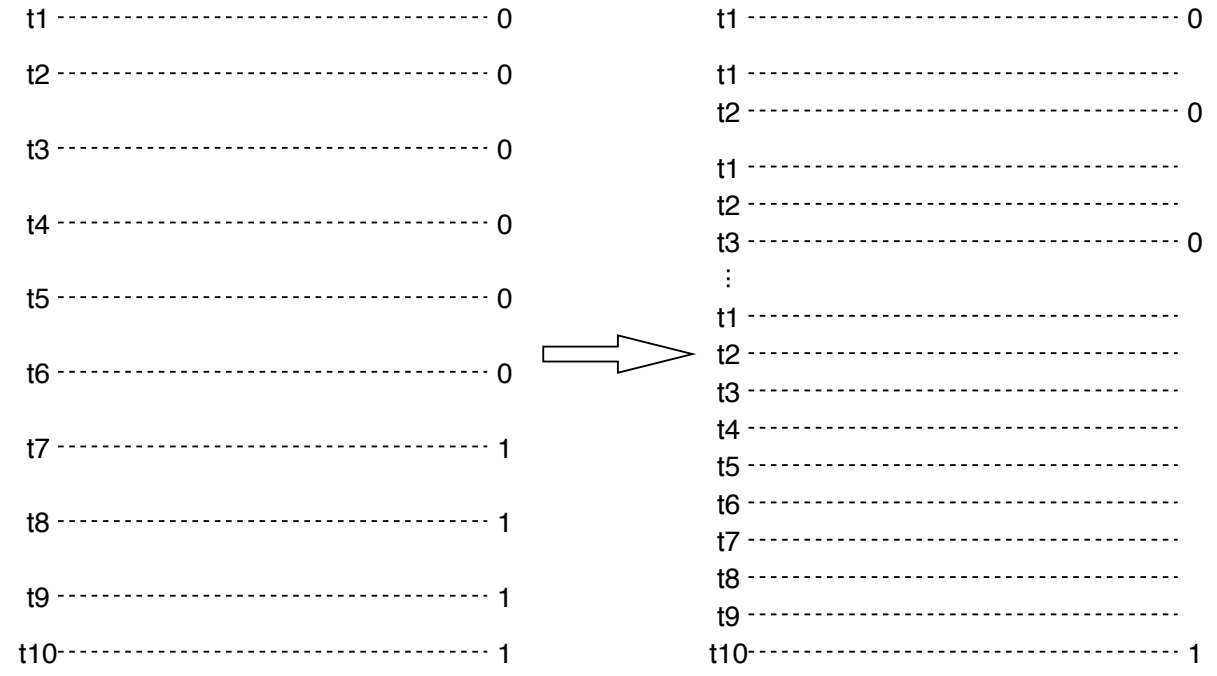


Figure 3.3: The left side shows the original data with sepsis, and the right side displays what the data look like after the processing.

Chapter 4

Experimental setup

To find the best implementation setting for the early prediction of sepsis, I conduct experiments for all the settings I mentioned in the data processing section (also shown in Table 6.1). The settings include combinations of two things: whether data standardization is performed or now, and if more variables are added. Furthermore, I train two models for each resulting setting and tune the number of epochs, as the best number of epochs differ for different models. Those two models, model 1 and model 2, are inspired from two published papers (Schellenberger et al., 2019; Roussel et al., 2019). They have been proved successful in predicting the PhysioNet/Computing in Cardiology Challenge 2019 data. More details on how the data is fed into the models and the specific configurations of the models are provided below.

After finishing the data pre-processing for all the 40336 PSV files, new instances are generated. I split the instances into batches. The batch size is set to 64 for Model 1 and to 256 for Model 2. The instances in the same batch are padded with zeros to the largest length of a record in the batch, before fitting them into the LSTM models. A masking layer is used first to account for the fact that the data has been padded to fit into the LSTM layers. Inspired by the LSTM model architecture drawn by Schellenberger et al., I use similar plots to present model 1 and model 2 architectures, as shown in Figure 4.1 and Figure 4.2.

Figure 4.1 shows the model 1 architecture, which includes two LSTM layers after the

masking layer and 400 hidden units in each layer. To avoid overfitting, a dropout value of 0.2 and a recurrent dropout of 0.5 are chosen for the LSTM layers. After the LSTM layers, four fully connected layers are added, in decreasing size of the number of hidden units in a layer. Specifically, the units are 250, 150, 100, and 50, respectively. The activation function used for those four layers is the Rectified Linear Unit (ReLu) function. The last layer is a Sigmoid dense layer, which outputs the predicted probabilities for the sepsis label. Batch normalization is included in model 1. The optimization is performed using the RMSProp optimizer, with a learning rate of 0.001.

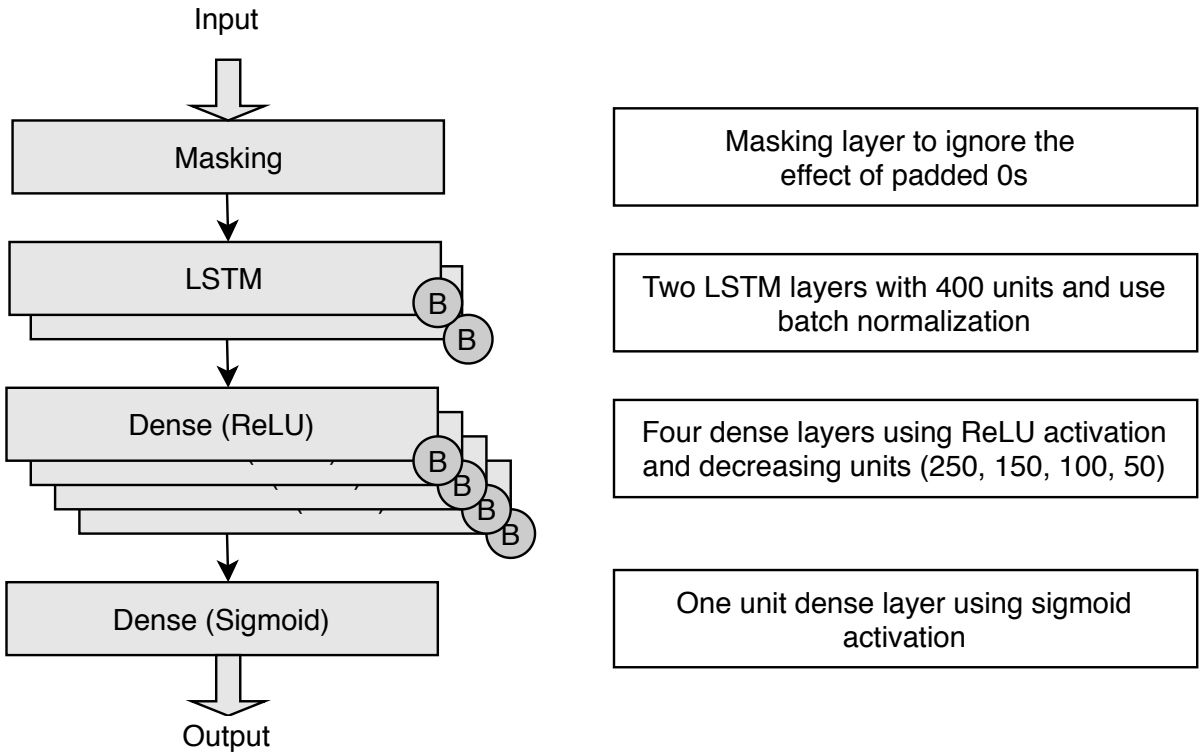


Figure 4.1: The left side displays the LSTM network architecture of Model 1. The boxes marked with B are batch normalization; The right side explains what is given in the corresponding layer.

Figure 4.2 displays the LSTM architecture of model 2. Model 2 includes three LSTM layers with 64 hidden units in each layer. The same dropout and recurrent dropout values as in Model 1 are used in the LSTM layers. Between the mask layer and the LSTM layer, three dense layers are included. The units vary based on the input layer size. The first layer's number of units is the input size multiplied by four; the second layer's number of units is

the input size multiplied by eight, while the third layer has a constant number of units, superficially 32. After the LSTM layers, three hidden dense layers are added. The number of units in those layers are 64, 512, and 4, respectively. The activation function used for all the layers is the Rectified Linear Unit (ReLU) function, except for the LSTM layers. At the last layer, a Sigmoid dense layer is included. Batch normalization and L2 regularization with a rate of 0.00001 are added in Model 2. The optimization is performed using the Adam optimizer with a learning rate of 0.0001.

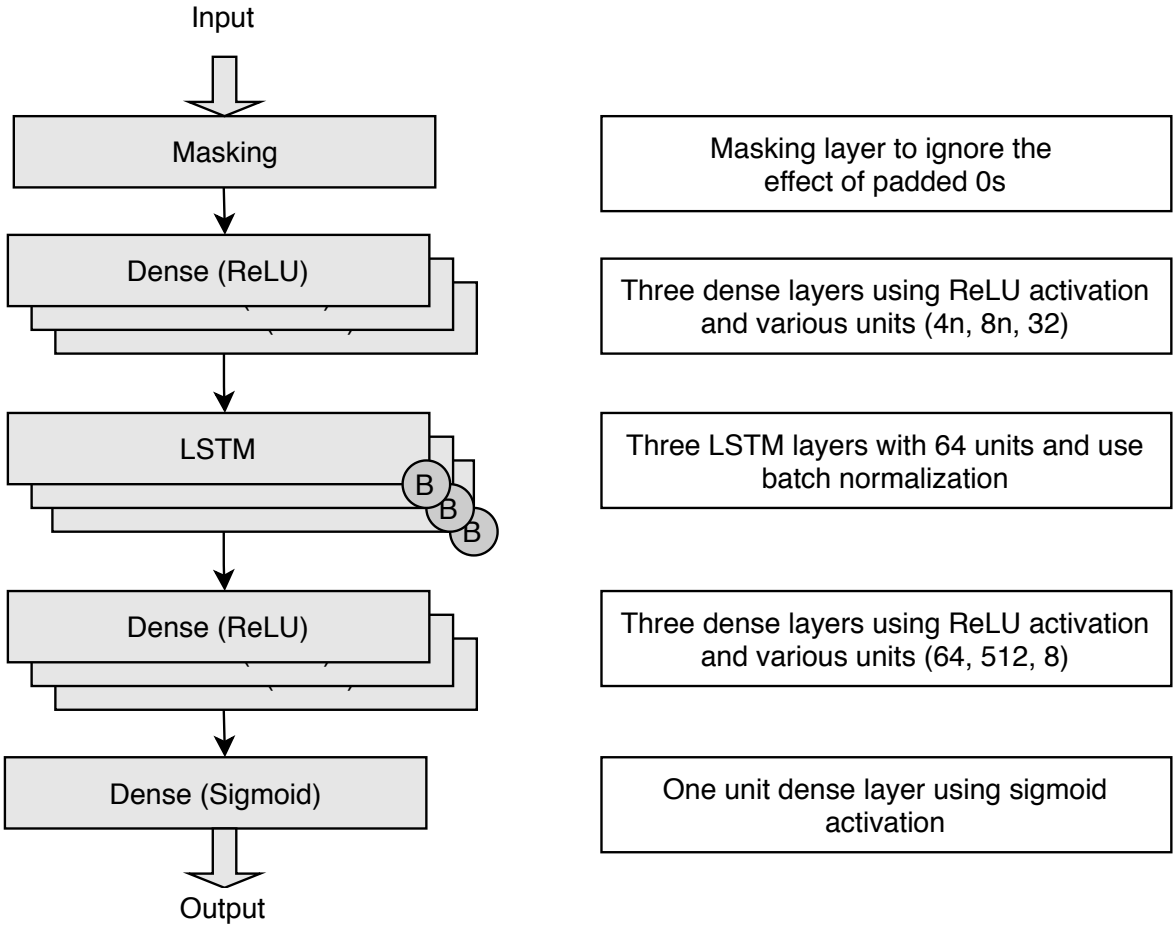


Figure 4.2: The left side displays the LSTM network architecture of Model 2. The boxes marked with B are batch normalization. The 'n' is the number of predictors, which can be 40 or 108; The right side explains what is given in the corresponding layer.

Both of the models are trained for five epochs. The reason for choosing five training epochs is that I observed that the performance obtained with 5 epochs is better than the performance using 10, 20, and 30 epochs. After finishing the training on the training data, I

evaluate the trained model on the validation data by calculating the utility score. The best threshold is selected by the criteria of generating a highest utility score using the validation data. Finally, I use the best threshold to compute the utility score for the test data.

Chapter 5

Implementation

The code is available at GitHub so that anyone with access to the PhysioNet/Computing in Cardiology Challenge 2019 data can reproduce my experiment. The experiment is conducted using Python programming language, the LSTM model is trained and evaluated with TensorFlow open-source library, and the code is organized in Jupyter Notebooks. I use the "Train_Val_Test_Generation.ipynb" file to split the whole dataset into three parts: training, validation, and test data. In this Jupyter notebook file, I specify the entire dataset's file path and the paths for storing the training, validation, and test data. For each implementation setting, I create two Jupyter notebooks: for the data cleaning and the LSTM model, respectively. I specify the file paths for storing the training, validation, and test data for both notebooks. In addition, the LSTM training file needs to define the file path for the Python evaluation file and the file paths for storing the prediction results. Finally, I implement the "compute_utility_score" function in the "LSTM" Jupyter Notebook file to compute the utility score using different thresholds. The function includes the Python file "evaluate_sepsis_score.py", which includes all the functions to compute " AUROC", "AUPRC", "Accuracy", "F-measure", and "Utility" values. The best threshold I select is the one that can obtain the highest utility score for the validataion data.

Adding 34 columns to the orginal data and deal with the missing value NaN with forward and backward filling with new added 34 predictors, the missing values in the orginal 40 predictors will be replaced with 0, for train, val and test

```
[ ]: for file in train_names:
         df= pd.read_csv(file)
         colNames = list(df)[:-7]
         for name in colNames:
             #print(name)
             df[name + '1'] = df[name]
             df[name + '1'].fillna(method='ffill',inplace=True)
             df[name + '1'].fillna(method='Bfill',inplace=True)
         df.fillna('0', inplace=True)
         df.to_csv(file, index=False,na_rep='NaN')
```

Figure 5.1: Sample code example showing one method used for data pre-processing.

Transform the train and val data to tensor data format and pad the data to make sure they have the same length (LSTM requirement)

```
[ ]: train_dataset = tf.data.Dataset.from_generator(train_generator,(tf.float32,tf.
      ↪float32))
     val_dataset = tf.data.Dataset.from_generator(val_generator,(tf.float32,tf.
      ↪float32))

     # padding for train and val data
     train = train_dataset.padded_batch(64,padded_shapes=([-1,-1],[]))
     val = val_dataset.padded_batch(64,padded_shapes=([-1,-1],[]))
```

Generate the model

```
[ ]: model = tf.keras.Sequential()
     model.add(tf.keras.layers.Masking(mask_value=0, input_shape=(None,40)))
     model.add(tf.keras.layers.BatchNormalization())
     model.add(tf.keras.layers.LSTM(units = 400, activation = 'tanh', dropout=0.2,␣
      ↪recurrent_dropout=0.5, return_sequences=True))
     model.add(tf.keras.layers.BatchNormalization())
     model.add(tf.keras.layers.LSTM(units = 400, activation = 'tanh', dropout=0.2,␣
      ↪recurrent_dropout=0.5, return_sequences=False))
     model.add(tf.keras.layers.BatchNormalization())
     model.add(tf.keras.layers.Dense(250,activation='relu'))
     model.add(tf.keras.layers.BatchNormalization())
     model.add(tf.keras.layers.Dense(150,activation='relu'))
     model.add(tf.keras.layers.BatchNormalization())
     model.add(tf.keras.layers.Dense(100,activation='relu'))
     model.add(tf.keras.layers.BatchNormalization())
     model.add(tf.keras.layers.Dense(50,activation='relu'))
     model.add(tf.keras.layers.BatchNormalization())
     model.add(tf.keras.layers.Dense(1,activation='sigmoid'))
```

Figure 5.2: Sample code showing the methods of data padding and model generation.

Chapter 6

Results

The experiments' results for the settings and models considered are shown in Table 6.1. As can be seen in the table, standardizing the data can produce better results for Model 1 than Model 2 when controlling the adding predictors' approach. In addition, the table shows that in terms of utility, adding predictors performs better than simply using the original 40 variables both when standardization is used or not used, for both Model 1 and Model 2. Furthermore, Model 1 obtains better overall results than Model 2 by controlling the standardization method and adding predictors' approach. Moreover, the best-chosen threshold using Model 2 is higher than Model 1 in my thesis. The best threshold values are all between 0.25 and 0.7, as shown in Table 6.1 and Table 6.2. However, the threshold can be as low as 0.07 if training the model with 30 epochs, as shown in the paper published by Schellenberger et al. (2019). Overall, the combination of no standardization, adding predictors, and using Model 1 produces the highest utility score, which means this combination produces the best overall early prediction of sepsis results.

Table 6.1: Prediction results for different settings and models. The "Standardization" column means whether I standardize the data before fitting into the model or not. The column "Adding Predictors" shows if extra predictors based on the original 40 predictors are added or not. The "Model" column represents which model is used in the training process. The performance is reported in terms of several metrics, including the area under the precision-recall curve (AUPRC), best threshold, F1 score, and utility score. The cells highlighted in green correspond to the setting with the best overall results among the eight implementation settings.

Standardization	Adding Predictors	Model	AUPRC	Best Threshold	F1 score	Utility Score
No	No	1	0.118	0.47	0.133	0.366
		2	0.079	0.57	0.113	0.321
	Yes	1	0.129	0.42	0.128	0.421
		2	0.085	0.73	0.124	0.343
Yes	No	1	0.112	0.61	0.116	0.361
		2	0.086	0.61	0.115	0.356
	Yes	1	0.093	0.45	0.128	0.385
		2	0.084	0.66	0.120	0.382

To avoid information overload in terms of graphs, I only report the results from the best and worse settings on the early prediction of the sepsis here, in Figures 6.1 and 6.2, respectively. The results of other settings are listed in Appendix A. The graphs show the area under the precision recall-curve (AUPRC), the area under the receiver operating characteristic curve (AUROC), the distribution of F1 scores, and the histogram of the prediction probabilities on the test data. The F1 score and utility score will be different if using a different threshold. The best threshold is the one which makes the utility score highest for the validation data. The precision values in Figure 6.1 (best scenario) start from a little above 0.4, resulting in an area under the precision-recall curve of only 0.13. However, in the worst scenario, the precision value starts from around 0.3, resulting in an AUPRC of only 0.08, as shown in Figure 6.2. The values of the true positive rate in Figure 6.1 are always above the red dashed line, and produce an AUROC value of 0.83, while the setting in Figure 6.2 only obtains an AUROC value of 0.78. Additionally, the F1 score in Figure 6.1 can get

around 0.2 by using a large threshold, while the best F1 score in Figure 6.2 is less than 0.2. Finally, the histogram of the predicted probabilities on the test data, shown in Figure 6.1 for the best setting, has more values below 0.1 than the histogram of predicted probabilities in the worst setting shown Figure 6.2. This shows that the best model predicts no sepsis labels with higher confidence. Overall, from the results of Figure 6.1 and Figure 6.2, we can conclude that even under the scenario of no standardization, implementing Model 2 along with not adding variables can worsen the prediction results significantly.

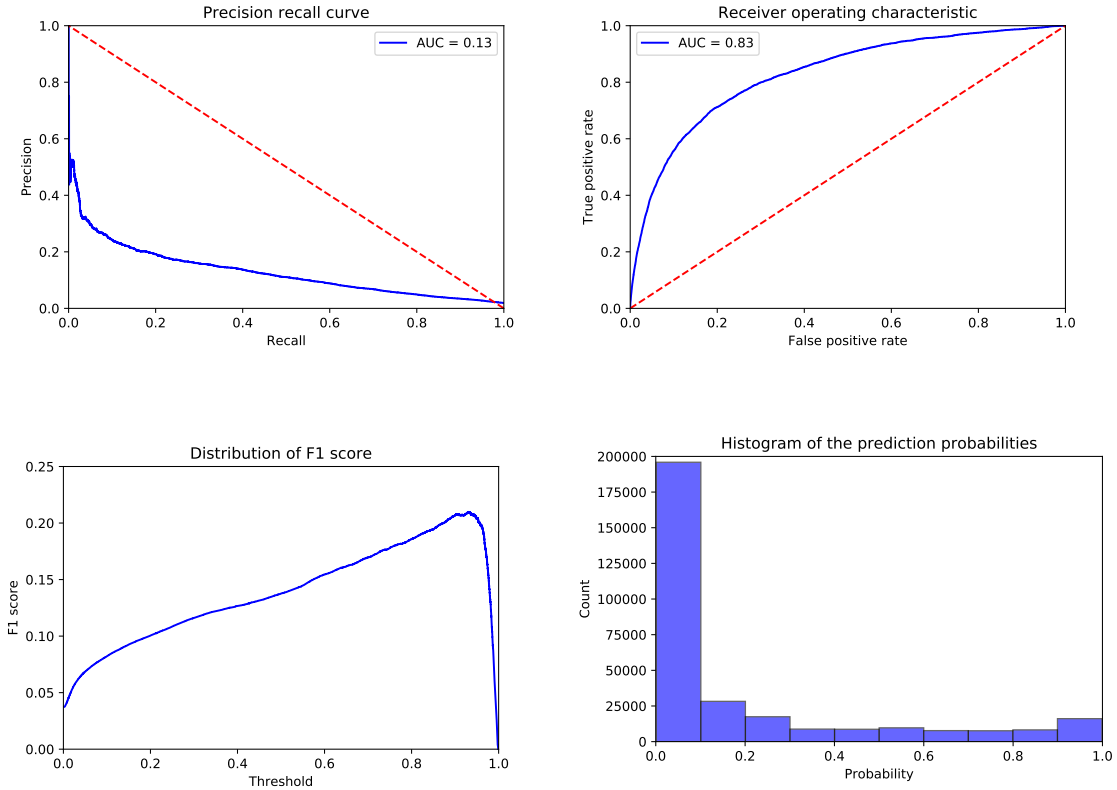


Figure 6.1: Results for setting 3. This setting uses no standardization, adds more predictors, and uses Model 1. The top left plot shows the precision-recall curve; the top right represents the receiver operating characteristic curve; the bottom left presents the distribution of F1 scores using different thresholds; the bottom right displays the histogram of the predicted probabilities on the test data.

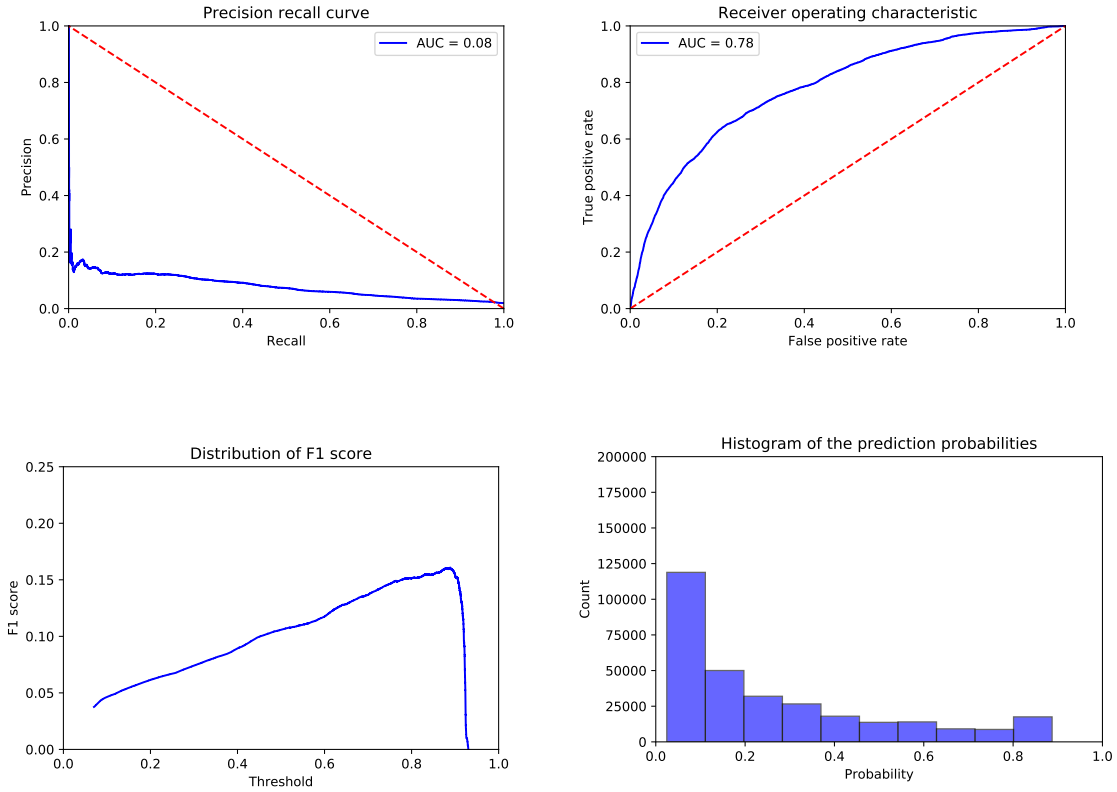


Figure 6.2: Results for setting 2. This setting uses no standardization, does not add predictors, and uses Model 2. The top left plot shows the precision-recall curve; the top right represents the receiver operating characteristic curve; the bottom left presents the distribution of F1 scores using different thresholds; the bottom right displays the histogram of the predicted probabilities on the test data.

Using the best implementation setting with no standardization, additional predictors added, and using Model 1, the early prediction of sepsis results varies if training the model with a different number of epochs. As shown in Table 6.2, as the number of epoch increases, the results become worse. Even though the F1 score does not change much by using a different number of epochs, the utility score and the AUPRC become smaller if using a larger epoch value. In addition, the best threshold also decreases as the number of epoch increases. Figure 6.3 shows the learning curves for training and validation loss. The pictures show that with the increase of the number of epochs, the loss becomes smaller. However, using five epochs achieves the highest utility score.

Table 6.2: Results for the best seeting using different number of epochs. The AUPRC stands for the area under the precision-recall curve. The best threshold is the one that achieves the highest utility score. The F1 and utility scores correspond to the best threshold. The cells highlighted in green correspond to the number of epocs that show the highest utility score.

Implementation Setting	Epochs	AUPRC	Best Threshold	F1 score	Utility Score
no standardization, adding predictors, using model 1	5	0.129	0.42	0.128	0.421
	10	0.109	0.40	0.130	0.392
	20	0.098	0.31	0.129	0.362
	30	0.087	0.25	0.127	0.341

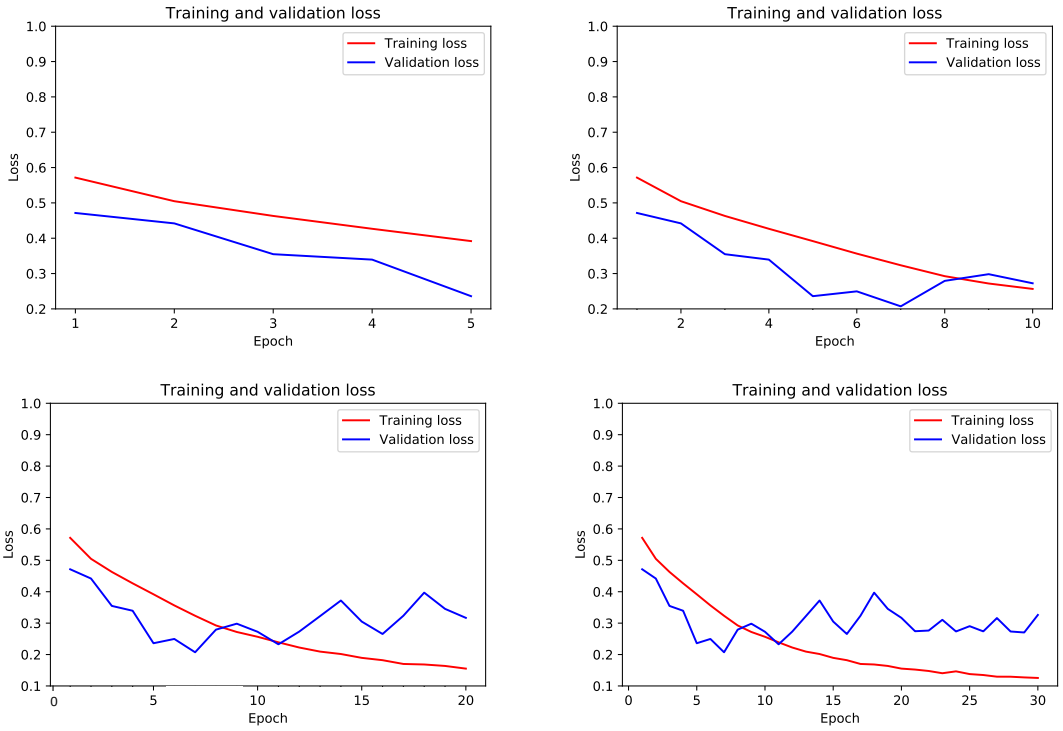


Figure 6.3: The learning curves show the training and validation loss using a different number of epochs. The top left shows the loss using five epochs; the top right displays the loss using ten epochs; the bottom left presents the loss using 20 epochs; the bottom right exhibits the loss using 30 epochs.

Figure 6.3 displays the training and validation loss using 5, 10, 20, and 30 epochs. The training loss becomes smaller with the increase of the number of epochs for the training data. However, the decreasing trends still hold for the validation data loss even though it does not change as much as the training data.

Chapter 7

Discussion

The results from the eight implementation settings, summarized in Table 6.1, show that whether standardizing data can improve the early prediction results depends on the model. More specifically, we can see that using Model 1 without standardization receives better outcomes than standardization, but that is not the case for Model 2. The reason could be that Model 2 has three dense layers to deal with the standardized data before the LSTM layers, making the data more appropriate for the LSTM layers than without standardization. However, further research should be conducted to prove this point. In addition, adding predictors presents better results for all the scenarios, which means the added extra predictors, such as the hourly differences, play an essential role in the prediction results. Moreover, Model 1 performs better than Model 2 when no standardization is performed and more predictors are added. The best implementation setting consisting of no standardization, adding predictors, and using Model 1 has the highest area under the precision-recall curve and utility score, but not the highest F1 score. The reason can be found when comparing Figures 6.1 and A.1. The histogram of the prediction probabilities in Figure 6.1 shows more than 185,000 of the predicted probabilities are less than 0.1, while Figure A.1 has around 150,000. Therefore, when using a small value of the threshold, the precision in Figure A.1 can arrive at around 0.8, which is higher than 0.5 in Figure 6.1. That is why the implementation of standardization, not adding predictors, and using Model 1 achieves a higher F1 score of 0.133.

Table 6.2 shows that using less epoch results in a higher utility score. With the increase of the training epochs, the results become worse. The reason is training the model with more epochs achieves less loss and larger accuracy for the training data. However, the utility score is the evaluation criterion in the early prediction of sepsis, instead of accuracy. In addition, the best threshold drops when increasing the number of epochs because the model predicts smaller probabilities after training longer.

From the learning curves of the training and validation loss in Figure 6.3, we can find the validation loss does not decrease smoothly. The validation loss becomes larger than training loss after training for more than 10 epochs. That is a clear sign that the model is overfitting. Even though this model helps me and Schellenberger et al. obtain acceptable prediction results. There could be a better LSTM model that can solve the overfitting problem.

Chapter 8

Conclusions and future work

Adding more predictors based on the existing variables can capture more critical information than using the original data. That information could play a vital role in the early prediction of sepsis, such as the differences between two adjacent hours. Besides, data scaling could destroy data diversity and weaken the strength of recognition. In general, as the number of epochs increases, the training loss will also decrease, and the prediction results become better. However, a small training loss may cause overfitting, meaning the model will not generalize well on the test data. Thus, it may be useful to sacrifice some loss and allow some false positive to avoid overfitting and improve the early prediction results on test data.

In conclusion, assuming that the goal is an early prediction of sepsis using LSTM, I recommend adding more predictors based on the original existing variables and not standardizing the data before fitting into the training model, using Model 1 discussed in this thesis. I suggest using a smaller value first and trying many different epochs for the number of training epochs until obtaining the best early prediction results.

However, further improvements may be demanded to produce more reliable outcomes. For example, concentrating on adding more variables based on the variables that are highly related to sepsis may obtain better results, such as "FiO_2" and "Platelets." In addition, I only try four different number of epochs and use two LSTM networks from existing published papers in this thesis. I assume that tuning more numbers of epochs and implementing

different LSTM model architectures should improve the early prediction results.

Bibliography

Desautels, T., Calvert, J., Hoffman, J., Jay, M., Kerem, Y., Shieh, L., Shimabukuro, D., Chettipally, U., Feldman, M. D., Barton, C., 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.

Fleuren, L. M., Klausch, T. L., Zwager, C. L., Schoonmade, L. J., Guo, T., Roggeveen, L. F., Swart, E. L., Girbes, A. R., Thoral, P., Ercole, A., et al. (2020). Machine learning for the prediction of sepsis: a systematic review and meta-analysis of diagnostic test accuracy. *Intensive care medicine*, pages 1–18.

Frost, R., Newsham, H., Parmar, S., and Gonzalez-Ruiz, A. (2010). Impact of delayed antimicrobial therapy in septic itu patients. *Critical Care*, 14(2):1–2.

He, Z., Chen, X., Fang, Z., Yi, W., Wang, C., Jiang, L., Tong, Z., Bai, Z., Li, Y., and Pan, Y. (2019). Early sepsis prediction using ensemble learning with features extracted from lstm recurrent neural network. In *2019 Computing in Cardiology (CinC)*, pages Page–1. IEEE.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Luan, Y. and Lin, S. (2019). Research on text classification based on cnn and lstm. In *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 352–355. IEEE.

Nejedly, P., Plesinger, F., Viscor, I., Halamek, J., and Jurak, P. (2019). Prediction of sepsis using lstm neural network with hyperparameter optimization with a genetic algorithm. In *2019 Computing in Cardiology (CinC)*, pages Page–1. IEEE.

Qiu, J., Wang, B., and Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one*, 15(1):e0227222.

Reyna, M. A., Josef, C., Seyedi, S., Jeter, R., Shashikumar, S. P., Westover, M. B., Sharma, A., Nemati, S., and Clifford, G. D. (2019). Early prediction of sepsis from clinical data: the physionet/computing in cardiology challenge 2019. In *2019 Computing in Cardiology (CinC)*, pages Page–1. IEEE.

Roussel, B., Behar, J., and Oster, J. (2019). A recurrent neural network for the prediction of vital sign evolution and sepsis in icu. In *2019 Computing in Cardiology (CinC)*, pages Page–1. IEEE.

Schellenberger, S., Shi, K., Wiedemann, J. P., Lurz, F., Weigel, R., and Koelpin, A. (2019). An ensemble lstm architecture for clinical sepsis detection. In *2019 Computing in Cardiology (CinC)*, pages Page–1. IEEE.

Seymour, C. W., Gesten, F., Prescott, H. C., Friedrich, M. E., Iwashyna, T. J., Phillips, G. S., Lemeshow, S., Osborn, T., Terry, K. M., and Levy, M. M. (2017). Time to treatment and mortality during mandated emergency care for sepsis. *New England Journal of Medicine*, 376(23):2235–2244.

Singer, M., Deutschman, C. S., Seymour, C. W., Shankar-Hari, M., Annane, D., Bauer, M., Bellomo, R., Bernard, G. R., Chiche, J.-D., Coopersmith, C. M., et al. (2016). The third international consensus definitions for sepsis and septic shock (sepsis-3). *Jama*, 315(8):801–810.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Vicar, T., Novotna, P., Hejc, J., Ronzhina, M., and Smisek, R. (2019). Sepsis detection in sparse clinical data using long short-term memory network with dice loss. In *2019 Computing in Cardiology (CinC)*, pages Page–1. IEEE.

Wang, Y., Xiao, B., Bi, X., Li, W., Zhang, J., and Ma, X. (2019). Prediction of sepsis from clinical data using long short-term memory and extreme gradient boosting. In *2019 Computing in Cardiology (CinC)*, pages Page–1. IEEE.

Appendix A

The results for other scenarios

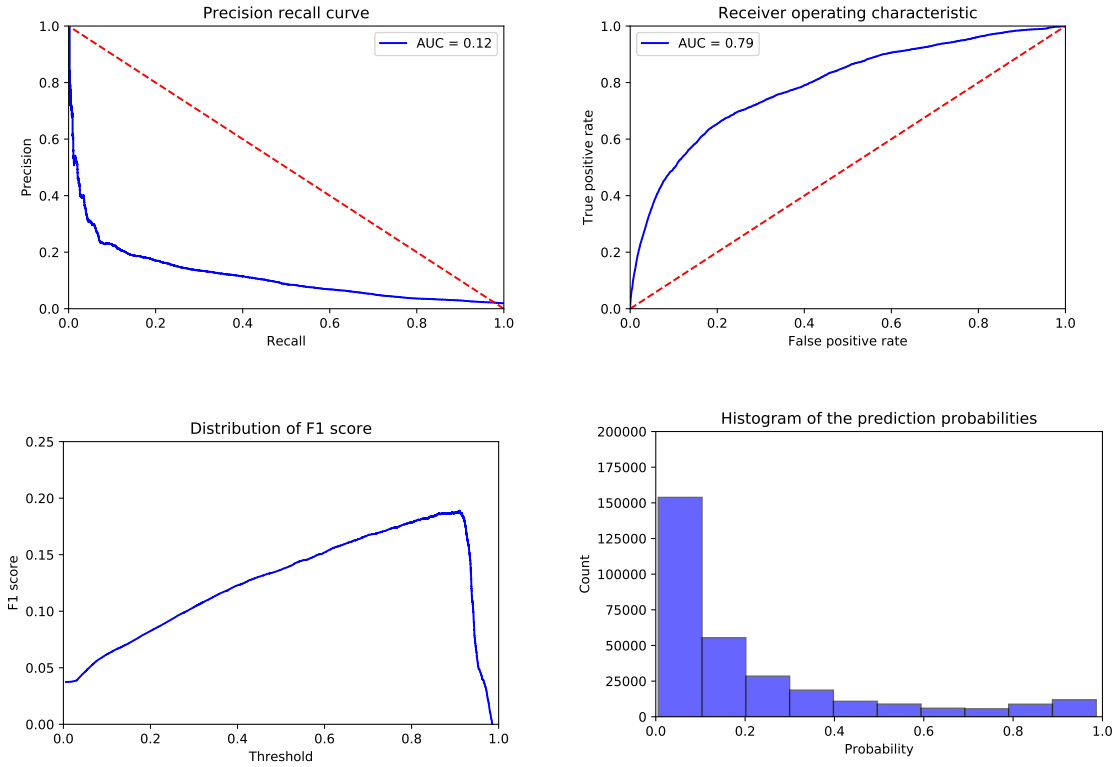


Figure A.1: Results for setting 1. This setting uses no standardization, does not add predictors, and uses Model 1. The top left plot shows the precision-recall curve; the top right represents the receiver operating characteristic curve; the bottom left presents the distribution of F1 scores using different thresholds; the bottom right displays the histogram of the predicted probabilities on the test data.

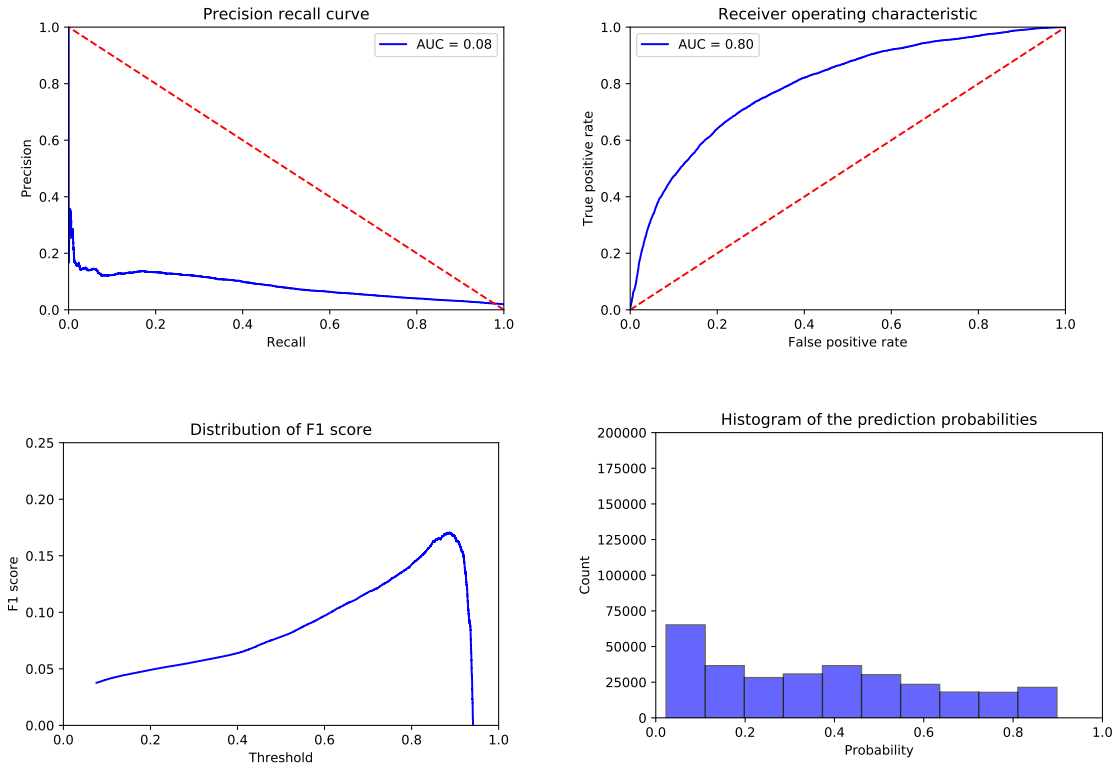


Figure A.2: Results for setting 4. This setting uses no standardization, adds predictors, and uses Model 2. The top left plot shows the precision-recall curve; the top right represents the receiver operating characteristic curve; the bottom left presents the distribution of F1 scores using different thresholds; the bottom right displays the histogram of the predicted probabilities on the test data.

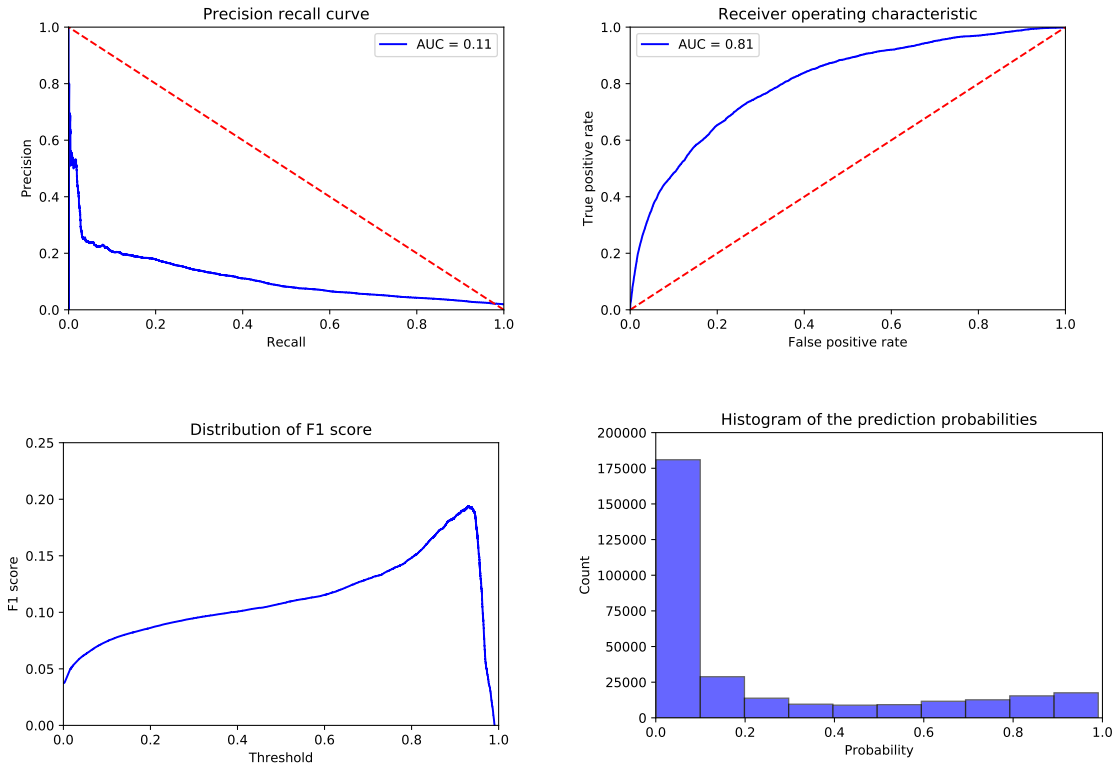


Figure A.3: Results for setting 5. This setting uses standardization, does not add predictors, and uses Model 1. The top left plot shows the precision-recall curve; the top right represents the receiver operating characteristic curve; the bottom left presents the distribution of F1 scores using different thresholds; the bottom right displays the histogram of the predicted probabilities on the test data.

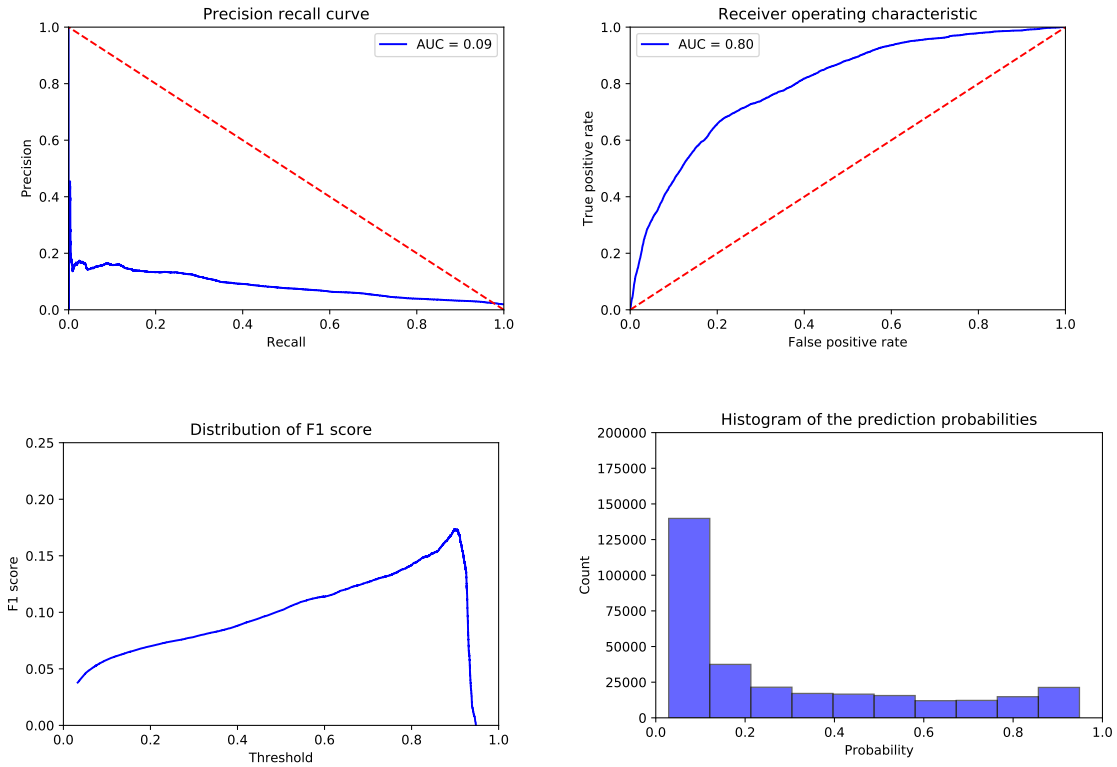


Figure A.4: Results for setting 6. This setting uses standardization, does not add predictors, and uses Model 2. The top left plot shows the precision-recall curve; the top right represents the receiver operating characteristic curve; the bottom left presents the distribution of F1 scores using different thresholds; the bottom right displays the histogram of the predicted probabilities on the test data.

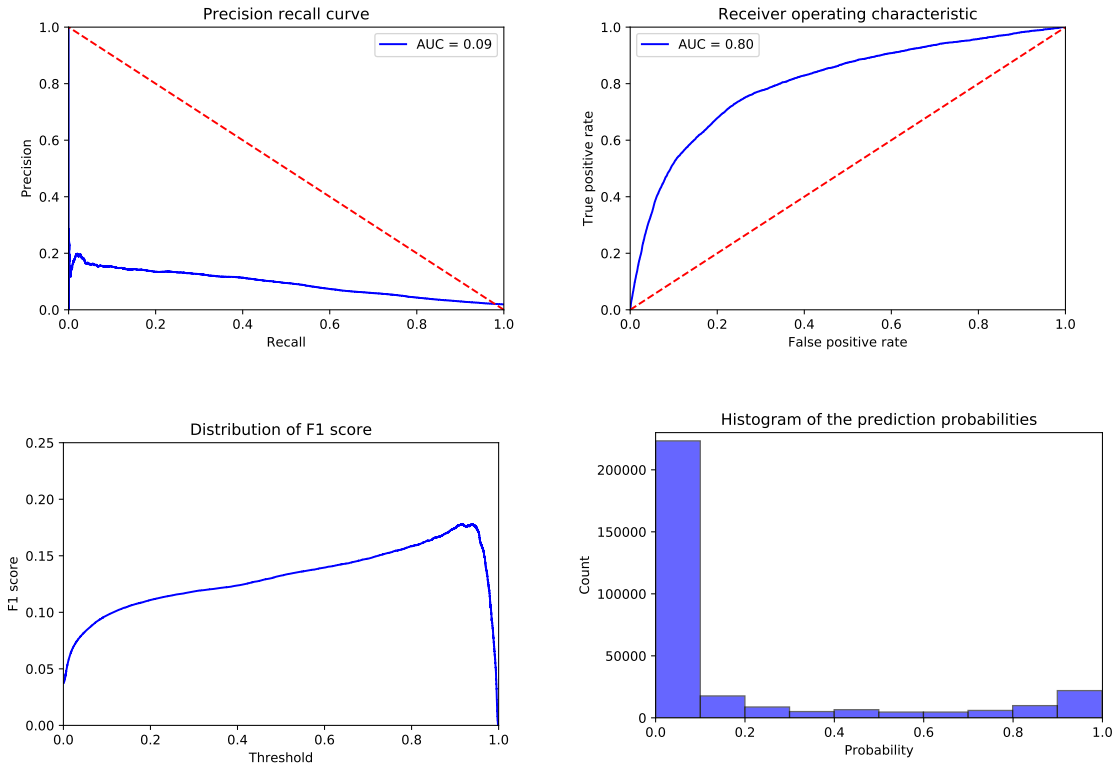


Figure A.5: Results for setting 7. This setting uses standardization, adds predictors, and uses Model 1. The top left plot shows the precision-recall curve; the top right represents the receiver operating characteristic curve; the bottom left presents the distribution of F1 scores using different thresholds; the bottom right displays the histogram of the predicted probabilities on the test data.

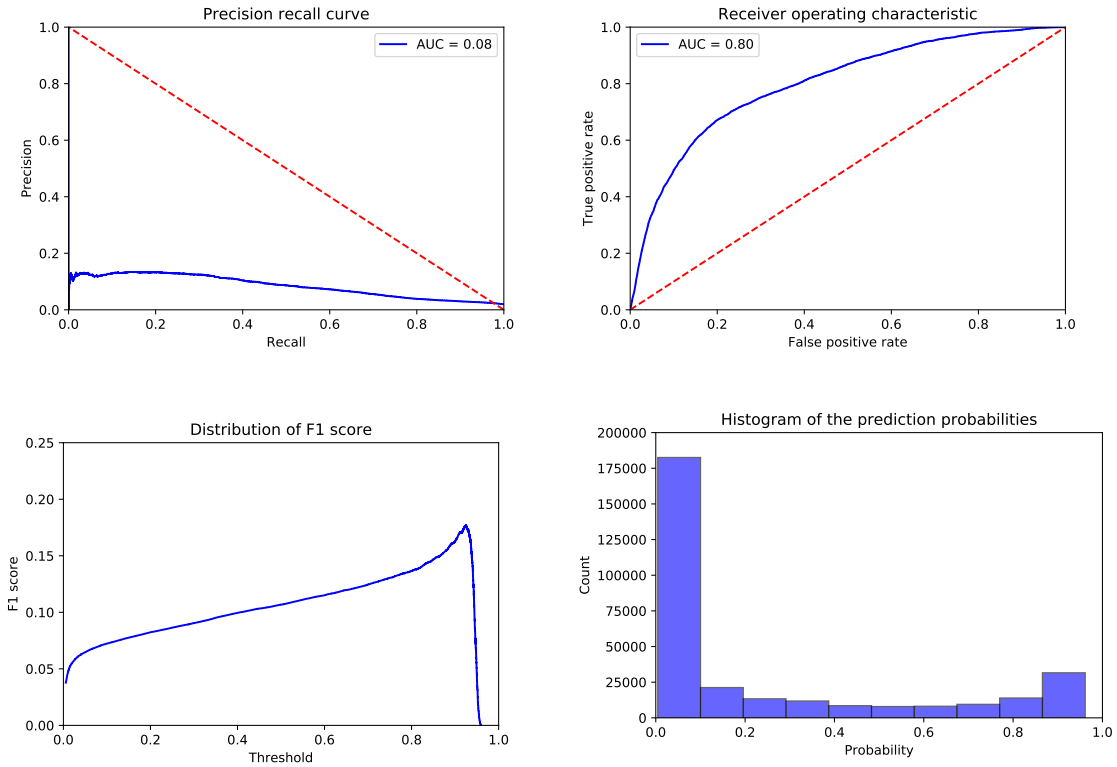


Figure A.6: Results for setting 8. This setting uses standardization, adds predictors, and uses Model 2. The top left plot shows the precision-recall curve; the top right represents the receiver operating characteristic curve; the bottom left presents the distribution of F1 scores using different thresholds; the bottom right displays the histogram of the predicted probabilities on the test data.