Lab in Data Analysis

Homework 1

Snir Lugassy (206312506)
Alon Shiri (323921940)

Repository: https://github.com/snirlugassy/sepsis_early_detection

Introduction

Sepsis is an extreme physiological response to an infection, with a toll of nearly 270,000 yearly deaths in the US alone. Our task was predicting Sepsis 6-hour before it occurs according to tabular data (PhysioNet 2019).

Our first approach was row-wise prediction while feeding and training RNN line-by-line, however, this approach yielded poor results and we had to seek a different solution (Gradient Boosted Trees).

Exploratory data analysis

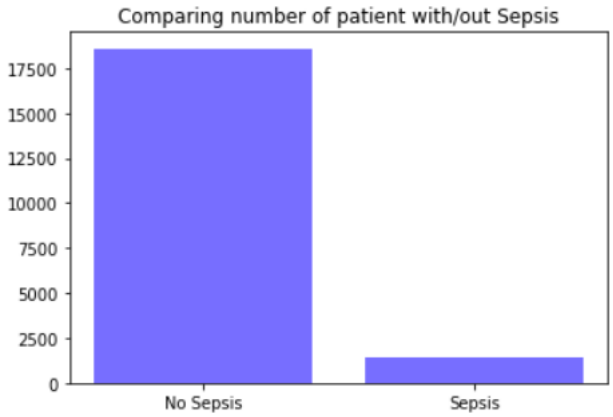
In the dataset there are 41 features (including the Sepsis target column), which can be divided as follows:

- 8 Vital signs features (e.g., Systolic blood pressure / Temperature)
- 26 Laboratory results features (e.g., pH / Chloride)
- 6 Demographic features (e.g., Gender / Age)
- 1 Target feature - SepsisLabel (1=the patient will have Sepsis in 6 hours)

We discarded the following features:

a. *Unit1 / Unit2*: Ignoring the unit in which the patient was administered (SICU / MICU) challenges our model to generalize better and be used in different hospitals.
b. *HospAdmTime*: we used the ICU length of stay measure instead (ICULOS).

The dataset is highly imbalanced with respect to the target feature:

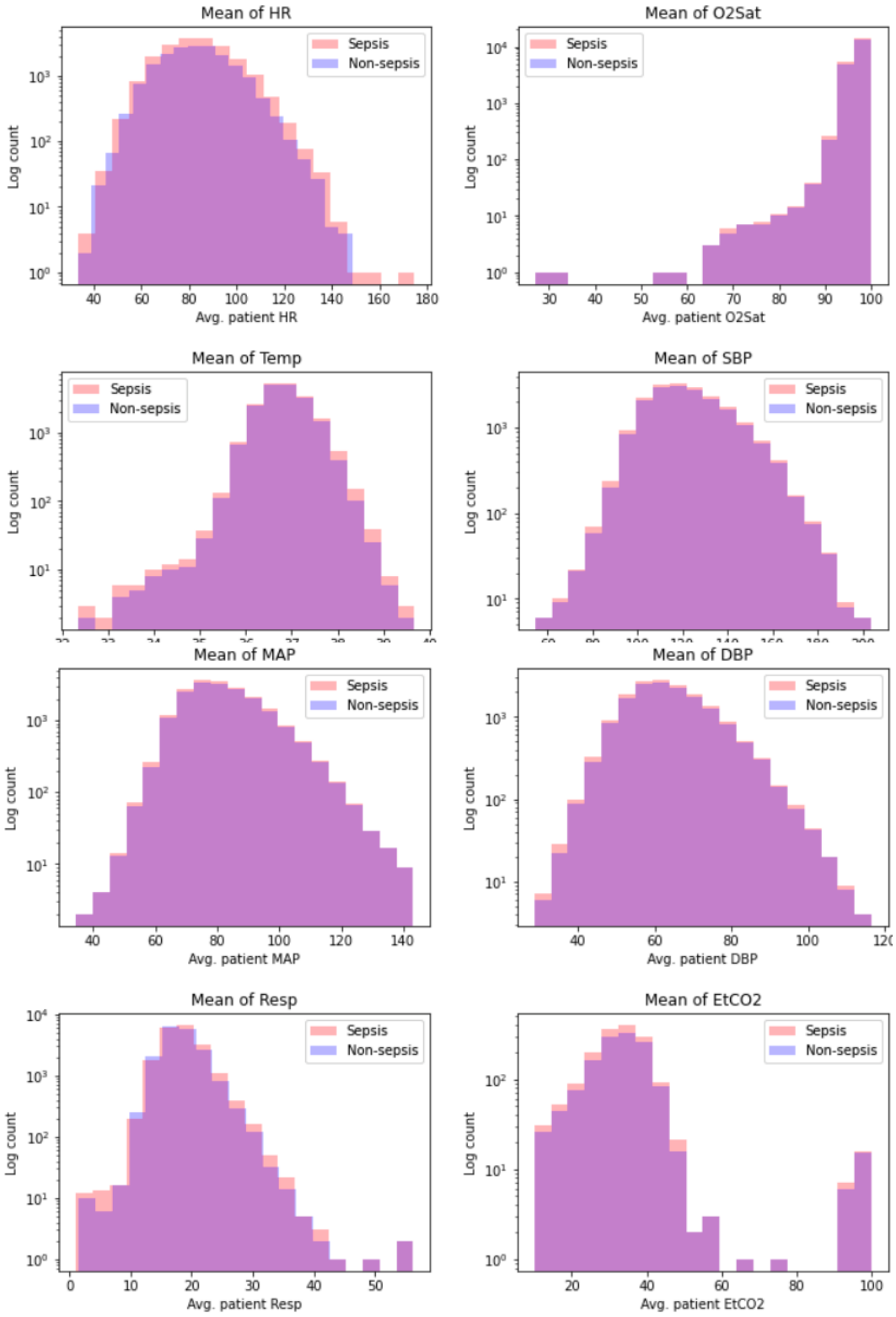


Our model will have to emphasize this imbalance and solve it, otherwise, it will be highly biased towards patients without Sepsis.

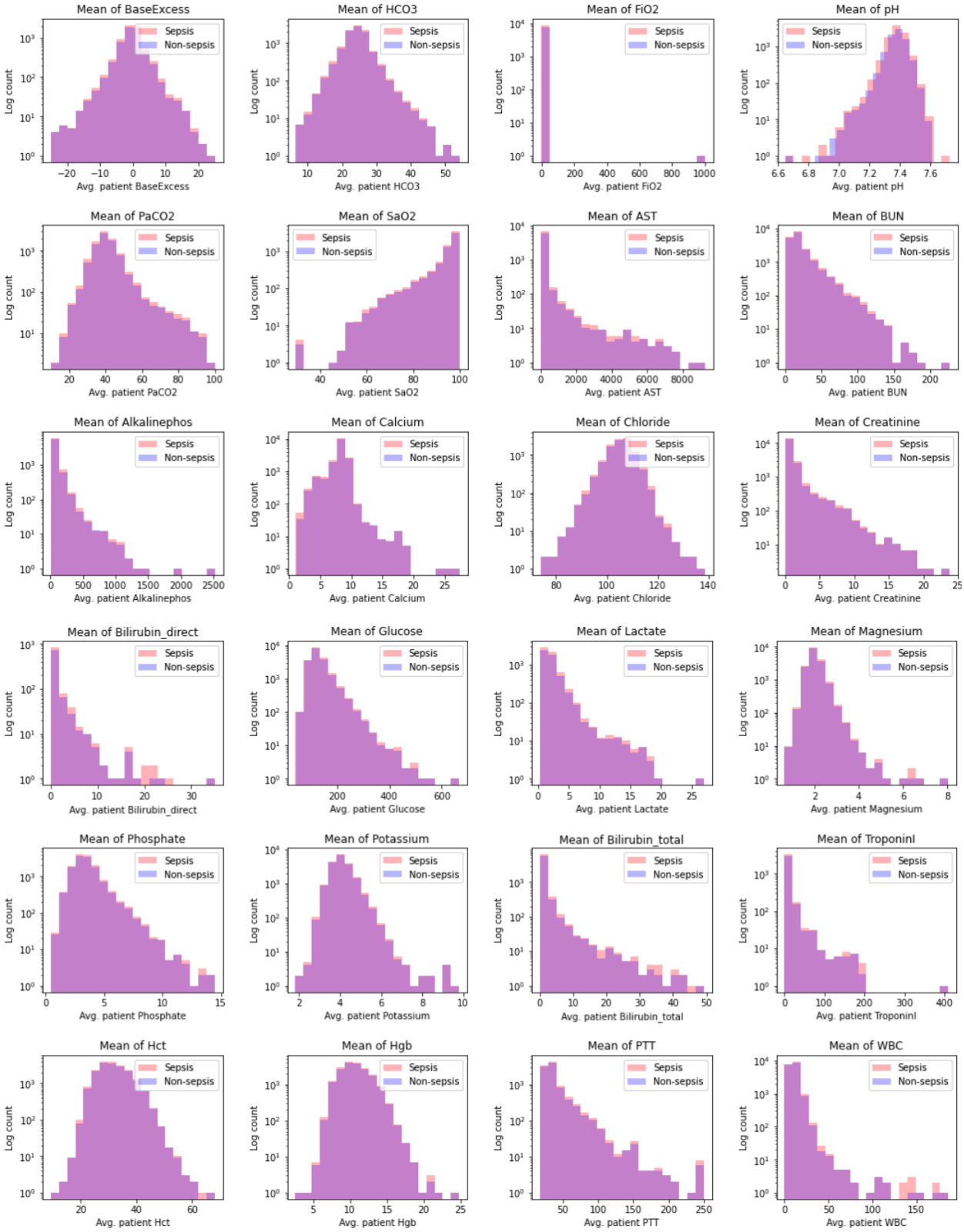
Features Distribution

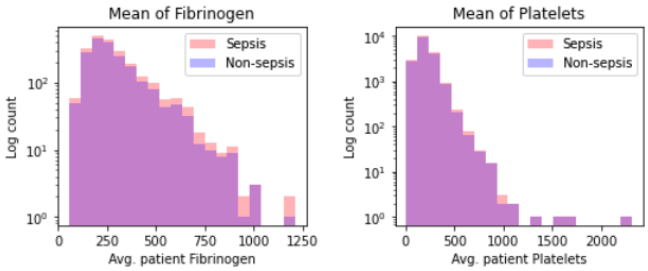
Most features have a large overlap in histograms between patients with Sepsis or without Sepsis, which indicates that classification based on those features themselves will not be efficient.

Vital signs features

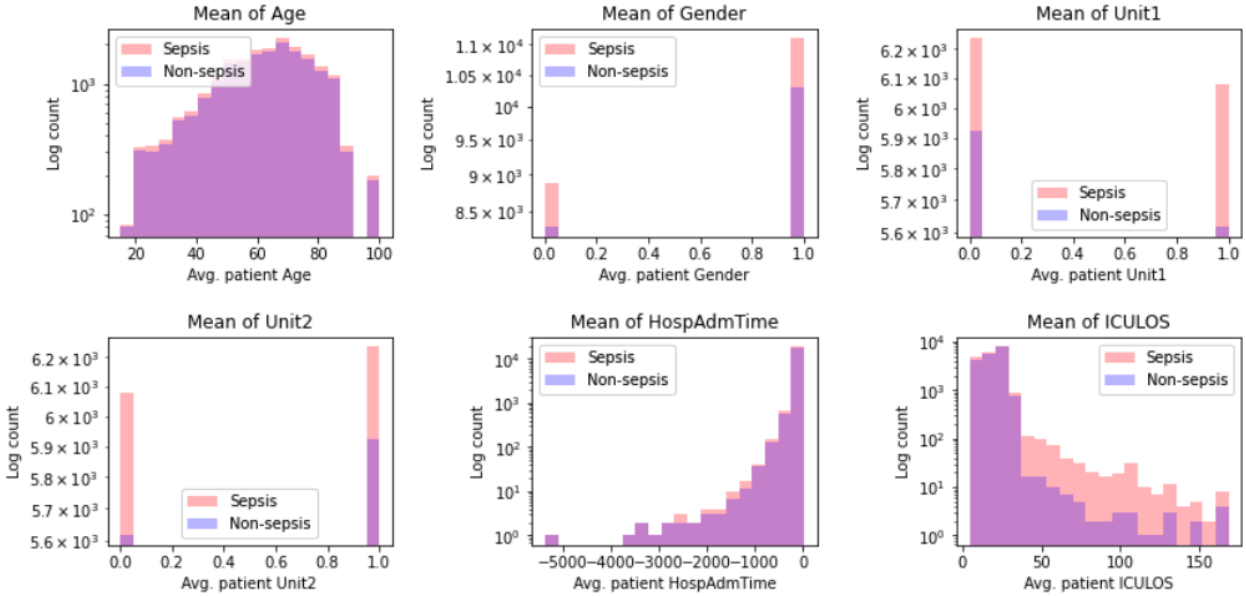


Laboratory features





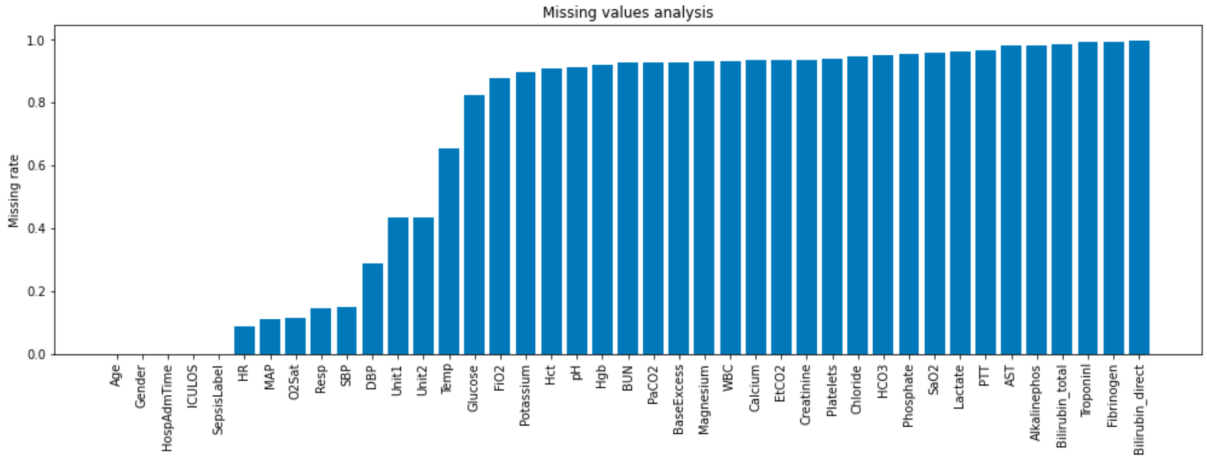
Demographics



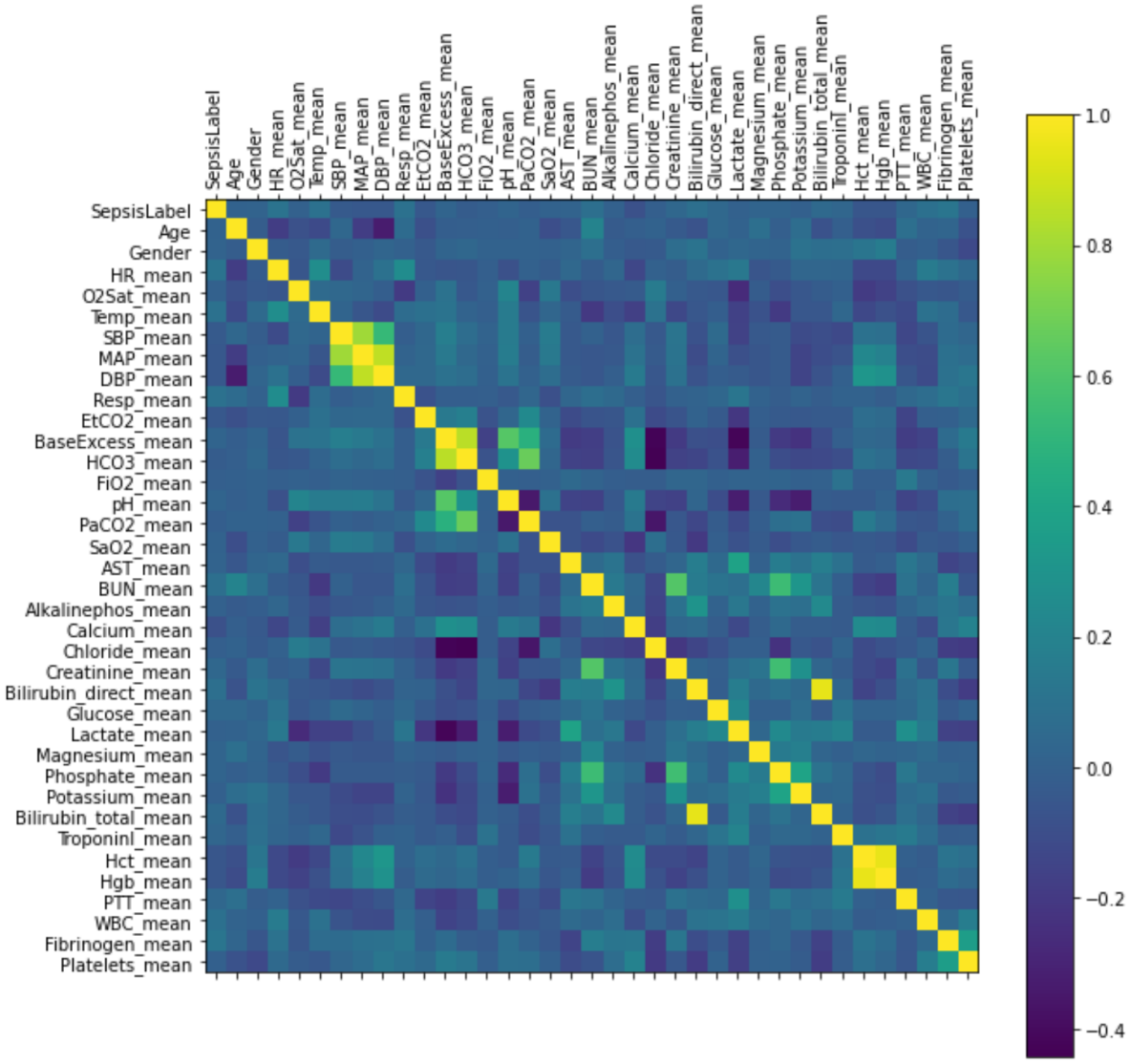
An interesting plot is the Avg. ICULOS (Bottom right, Average length of stay in the ICU unit). We can observe that patients with Sepsis tend to have a longer stay in the ICU unit.

Missing data

The given dataset is X with missing values, especially for the laboratory results features (more than 80% missing on average). The following graph plots the missing rate of each feature:



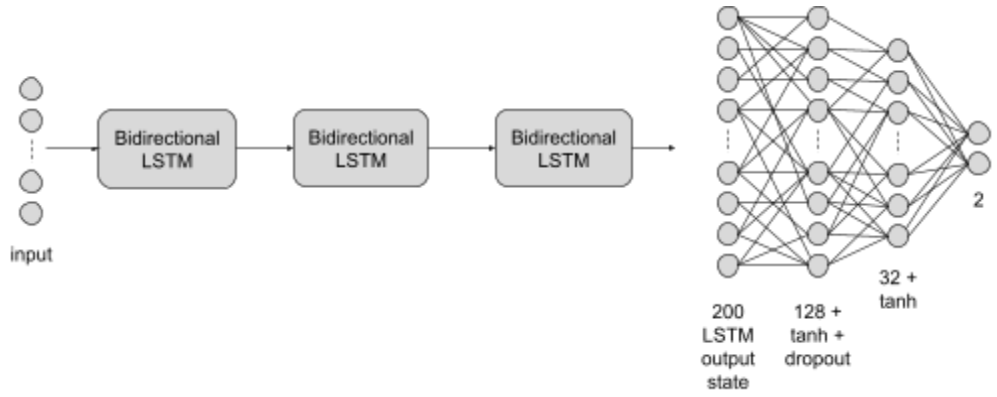
Features Correlation Matrix



We observe that the SepsisLabel is not correlated strongly with other features, however, there is a slightly larger correlation between SepsisLabel and HR/Temp/Resp.

The first approach (LSTM)
Our initial approach was using RNN (LSTM), feeding the model line-by-line, in order to predict row-wise (hourly) the SepsisLabel according to all previous rows (entire hospitalization measurements).

Architecture



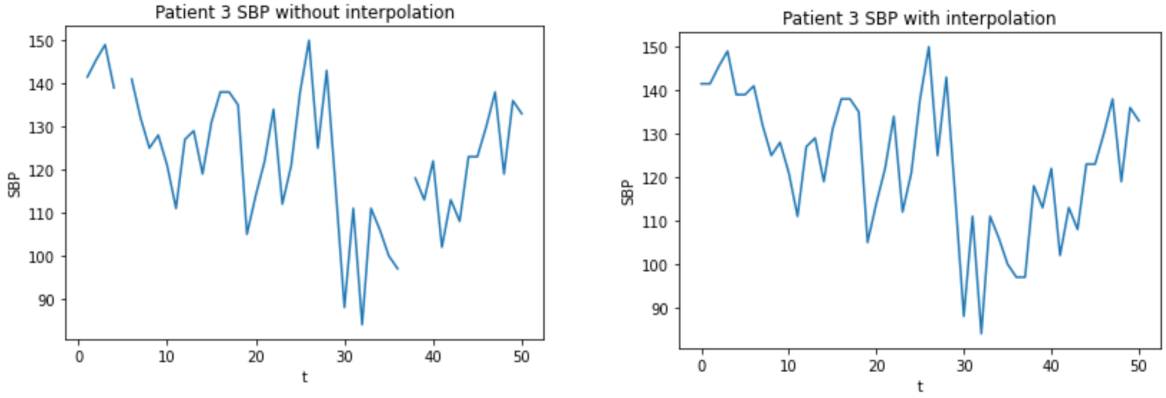
Since deep learning approaches can't handle missing values as input (as far as we are concerned) we had to select only features with low missing rates and impute them if necessary.

In order to avoid dropping/imputing a substantial amount of data, We considered only features with a missing rate lower than 0.2 (justified by Figure XXX), which resulted in the following 9 features:

1. Age
2. Gender
3. Temp
4. HR
5. Resp.
6. SBP
7. MAP
8. ICULOS (used instead of HospAdmTime)
9. O2Sat

We used SciPy's interpolation method for 1D vectors "interp1d" in order to impute missing values of the vital signs. In the rare case of a patient who completely missed the entire column, we excluded him from the training set.

Imputation example using “interp1d” with the “nearest” method:



We used the “nearest” interpolation method after empirically testing several methods, “nearest” seemed to be more locally concerned and smoother for this type of time-series data.

Performance

The RNN approach resulted in poor performance (very overfitted and F1 score < 0.5), we assume one or more of the following:

1. The RNN wasn’t trained for long enough.
2. There is not enough data for training (only 20,000 patient vectors).
3. The unbalanced data was causing the network to prefer 0 as output.
4. Our architecture wasn’t suitable.

Since we had a short time to solve this task, we moved on to different approaches that discard the element of time in the data, and summarize each patient into a single vector, then we could use more classical classification approaches.

The second approach (XGBoost)

In our second approach we processed each patient's data considering the following features transformation:

- For each dynamic feature X:
    - Mean of X
    - Range of X
    - The standard deviation of X
- For each vital sign feature X:
    - Maximum of X
- Maximum of “SepsisLabel”, “Gender”, “Age”, and “ICULOS”.

**Dynamic feature = vital sign / lab result*

The above processing resulted in 59 features (including the target) and was trained using Extreme Gradient Boosting (XGBoost) using the “log-loss” evaluation metric for 100 rounds.

In contrast to Neural networks or other classification methods, Decision trees can handle missing data quite well, therefore, besides removing highly missing features, we didn’t handle missing values any further.

Moreover, The usage of boosted trees allows us to retrieve and visualize important features, since the decision trees hold information about the importance of each feature. Generally, the more important the feature is the more it will be used to make key decisions and improve performance while training.

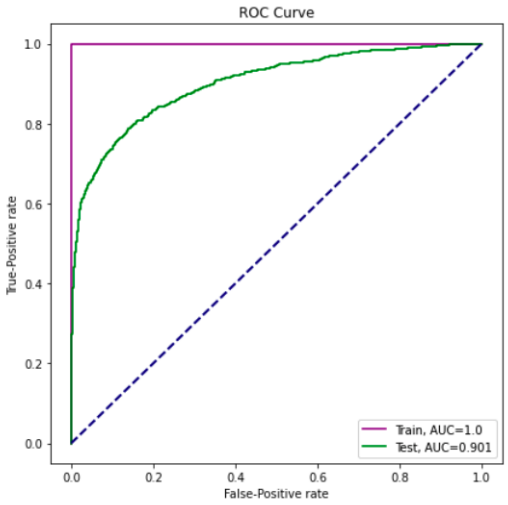
In order to tackle the imbalanced data we weighted the training sample as follows (passed to XGBoost as a vector of weights):

$$w_x = \alpha \cdot y + (1 - \alpha)(1 - y)$$

$$\alpha = 0.7$$

Without weighting the samples the F1-score was nearly zero, since the model was overfitted towards negative Sepsis samples.

Performance

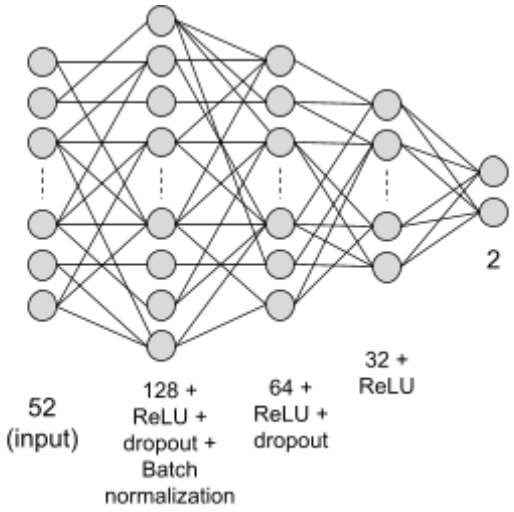


Train F1-Score: **0.99**
Test F1-Score: **0.624**

The above score was achieved after manual tweaking of the classifier’s parameters in order to maximize the test F1-Score.

Third approach (Neural Network with patient summarized vectors)
Our first attempt using Deep learning (LSTM) wasn't successful, but using the summarized patient feature vectors in the second approach was quite promising, therefore, our forth approach uses the same feature generation process as in the second approach and trains a fully connected neural network. We hadn't used the first order interactions since the network should learn the optimal interactions by itself.

Architecture



We used the Cross-entropy loss function with the following weighting to balance the positive Sepsis instances:

$$w_x = 5 \cdot y + 1 \cdot (1 - y)$$

Performance
Test accuracy: **0.905**
Test F1-Score: **0.455**

After trying several modifications both on the hyperparameters and the network, we couldn't achieve better results, with more time for submission we would try:

- Larger feature generation process
- More training iterations (epochs)
- 1D CNN over important features (e.g., 1D CNN over the O2Sat)
- Solving the data imbalance with more techniques (e.g., data generation)
- Solving the missing data with better techniques (e.g., KNN imputation / column-wise regressors)

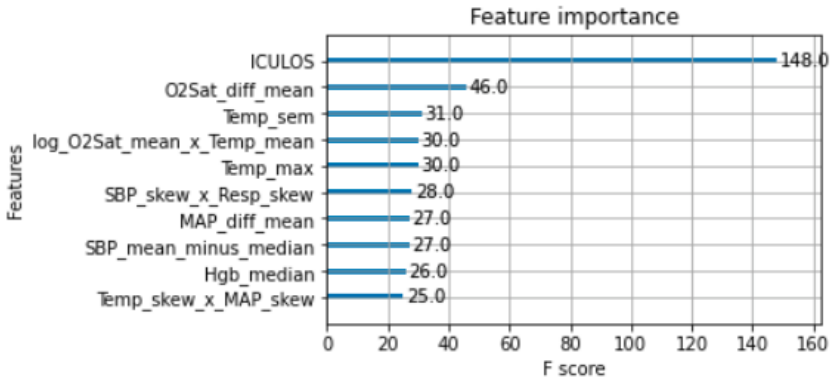
Fourth approach (XGBoost + better feature generation)
Since the gradient boosted trees were very simple to build and train, we used XGBoost again but tried loading more features into the model in order to improve performance.

We used the following feature extractions:

- For each dynamic feature X:
    - Mean of X
    - Skewness of X
    - The rolling mean of X with a window size of 3
    - Row-wise difference mean of X
    - Median of X
    - Mean minus the Median of X
    - Range of X
    - The standard deviation of X
    - Standard error of X
- For each dynamic feature X and Y (interactions):
    - The logarithm of X times Y
    - Mean of X over mean of Y
    - The skewness of X times the skewness of Y
- For each vital sign feature X:
    - Maximum of X
- Maximum of “SepsisLabel”, “Gender”, “Age”, and “ICULOS”.

This resulted in 485 features (some are highly sparse)

The XGBoost library provides us with a built-in feature importance score for each feature, after training our boosted trees the top 10 important features:



The most dominating feature in the classification decision of the boosted trees is the length of stay in the ICU unit (ICULOS), which settles with our prior that the more people stay in the ICU the worse their medical condition is / will be. Moreover, many other generated features are also included in the top 10, which hints that our feature generation process was beneficial for the model.

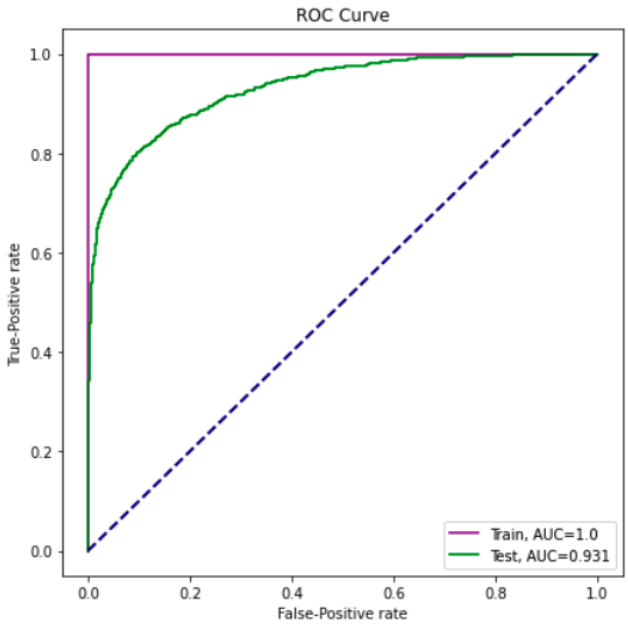
Sample Weights

Similar to the previous XGBoost, In order to tackle the imbalanced data we weighted the training sample as follows (passed to XGBoost as a vector of weights):

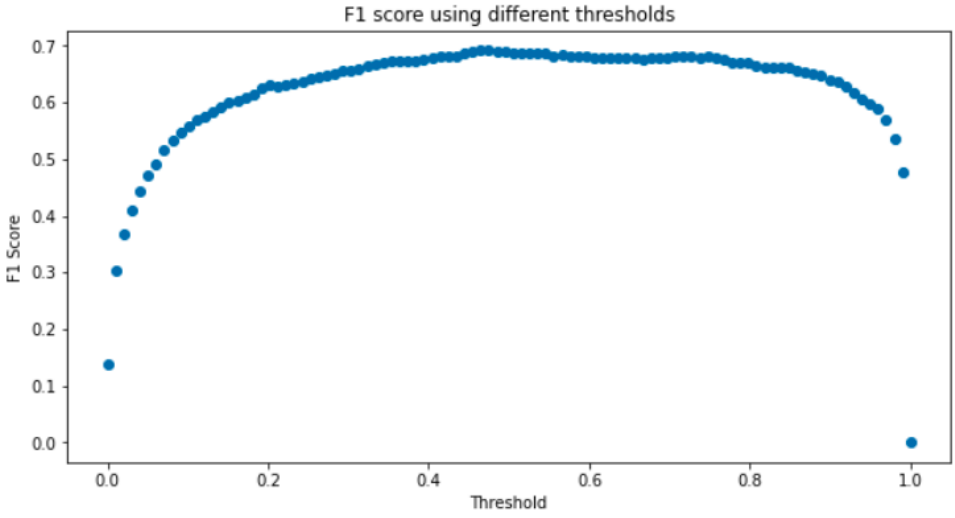
$$w_x = \alpha \cdot y + (1 - \alpha)(1 - y)$$

$$\alpha = 0.92$$

Performance



Since we are interested in the best F1-score, we chose the classification threshold which maximizes this metric (can be done using any evaluation metric).



Train F1-Score: **0.992**
Optimal test F1-Score: **0.691**

This model outperformed all other models we have tried so far.

Summary

Throughout this project, we have learned about solving the same task using different approaches, data cleaning, processing time-series data, and squeezing models in order to get the best performance.

Moreover, we have learned that although deep learning techniques are popular and a go-to tool in the data scientist toolbox (and for a good reason), it takes time to construct the network correctly and achieve good results. While methods like XGBoost achieved relatively great results without too much data processing and missing data handling.

In order to further build this model and achieve SOTA performance, we would fix and improve the RNN model since the element of time is critical for this task in the real world. (e.g., for model deployment in medical devices / medical center).