Lab II - HW1

Yoav Sgan-Cohen - 318679354
Vladimir Shulman - 208748475
Ido Yutko - 212765705

1. Executive summary

In this lab we aimed to develop a predictive model capable of identifying sepsis in patients using sequential patient data. Sepsis is a life-threatening condition that occurs when the body's response to an infection spirals out of control, potentially leading to organ dysfunction, tissue damage, and even death. Early detection and intervention are crucial for improving patient outcomes, making the development of an accurate and timely predictive model a valuable tool for healthcare professionals.

We approached this problem by selecting and evaluating 4 different machine learning models, with a focus on comparing their performance using the F1 score, precision and recall scores. The F1 score is a balanced measure of a model's precision and recall, providing a comprehensive assessment of its ability to accurately identify sepsis cases while minimizing false positives and false negatives.

2. Exploratory Data Analysis

a. The dataset includes various features that can be divided into four groups for better understanding:
    i. Group 1: Vital Signs
        1. HR: Represents heart rate measured in beats per minute
        2. O2Sat: Denotes pulse oximetry, measured in percentages
        3. Temp: Represents temperature, measured in degrees
        4. SBP: Stands for systolic blood pressure
        5. MAP: Represents mean arterial pressure
        6. DBP: Denotes diastolic blood pressure
        7. Resp: Indicates respiration rate, measured in breaths per minute
        8. etCO2: Stands for end-tidal carbon dioxide

    ii. Group 2: Laboratory Measurements
        1. BaseExcess: Measures the amount of excess bicarbonate

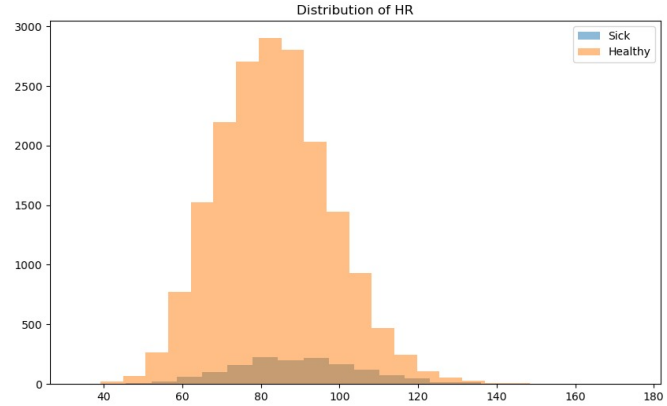
2. HCO3: Represents bicarbonate levels
3. FiO2: Stands for the fraction of inspired oxygen
4. pH: This feature only contains missing values
5. PaCO2: Represents the partial pressure of carbon dioxide in arterial blood
6. SaO2: Indicates oxygen saturation from arterial blood
7. AST: Stands for aspartate transaminase
8. BUN: Represents blood urea nitrogen
9. Alkalinephos: Denotes alkaline phosphatase
10. Calcium: Refers to calcium levels in the blood
11. Chloride: Indicates chloride levels in the blood
12. Creatinine: Measures creatinine levels in the blood
13. Bilirubin_direct: Shows the efficiency of liver function
14. Glucose: Represents blood glucose levels
15. Lactate: Measures lactic acid levels in the blood
16. Magnesium: Indicates magnesium levels in the blood
17. Phosphate: Represents phosphate levels in the blood
18. Potassium: Denotes potassium levels in the blood
19. Bilirubin_total: Similar to feature 13 but uses different scales
20. TroponinI: Measures troponin levels, essential for muscle function
21. Hct: Represents hematocrit levels in the blood
22. Hgb: Denotes hemoglobin levels in the blood
23. PTT: Stands for partial thromboplastin time
24. WBC: Represents white blood cell count
25. Fibrinogen: Measures fibrinogen levels, indicating liver function
26. Platelets: Shows platelet levels, crucial for blood clot prevention

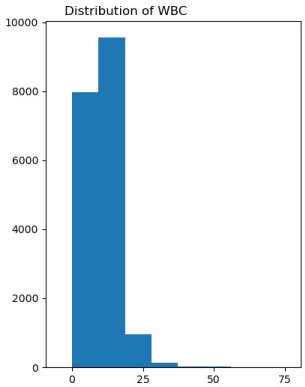
iii. Group 3: Demographic Information
1. Age: Indicates the patient's age
2. Gender: A binary variable with 0 representing female and 1 representing male
3. Unit1: Administrative identifier for the Medical Intensive Care Unit (MICU)
4. Unit2: Administrative identifier for the Surgical Intensive Care Unit (SICU)
5. HospAdmTime: Time in hours between hospital admission and ICU admission
6. ICULOS: Duration of ICU stay in hours since admission

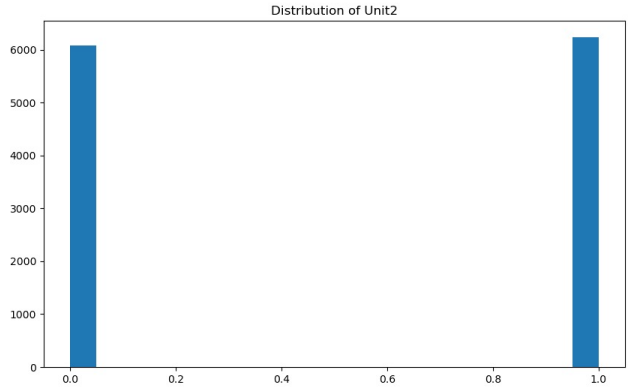
iv. Group 4: Sepsis Indication
1. SepsisLabel: A binary variable with '1' indicating the presence of sepsis or a record less than 6 hours before the first sepsis indication. It is '0' otherwise.

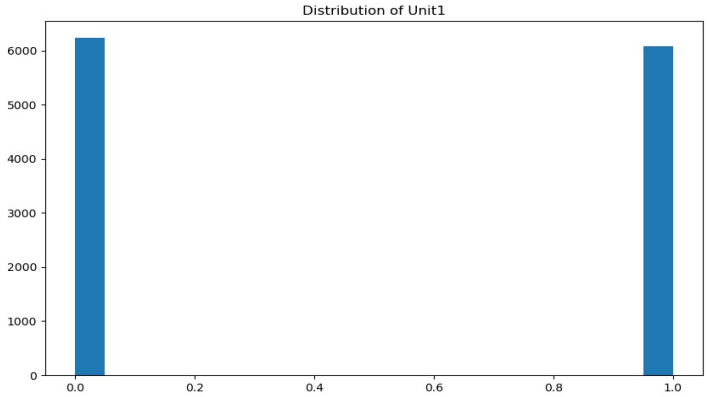
b. Inspecting the features distribution, comparative analysis between features (with plots and hypothesis testing):
First we averaged for each patient the values per feature over all of its features through the time recorded in the data.
After that, we aggregate all of the patients into one Pandas dataframe where each patient has a vector of features (the average of its values as we said before).
We had hypothesis that there is positive correlation between the measured HR and the existence of Sepsis, i.e. patients with Sepsis has higher measurements of HR.

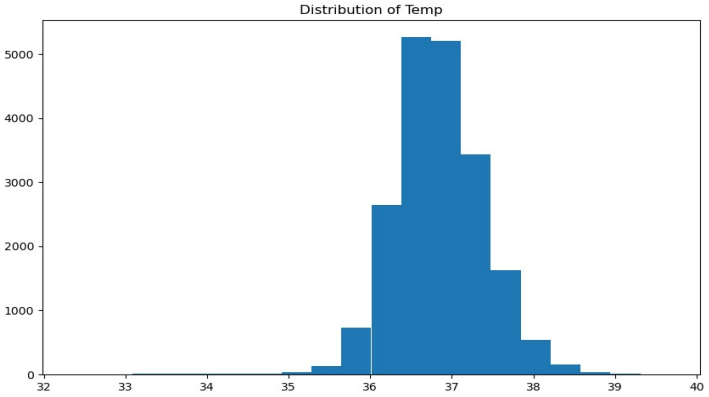
We did a hypothesis testing: our H0: The expected value of HR is the same on the patients group that have sepsis and the patients group that not have it
We got a small P Value of: 1.8806e-49

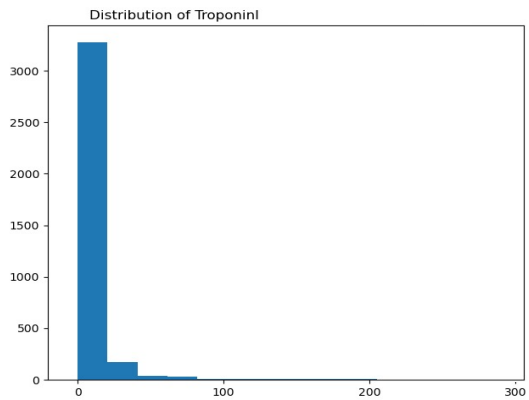


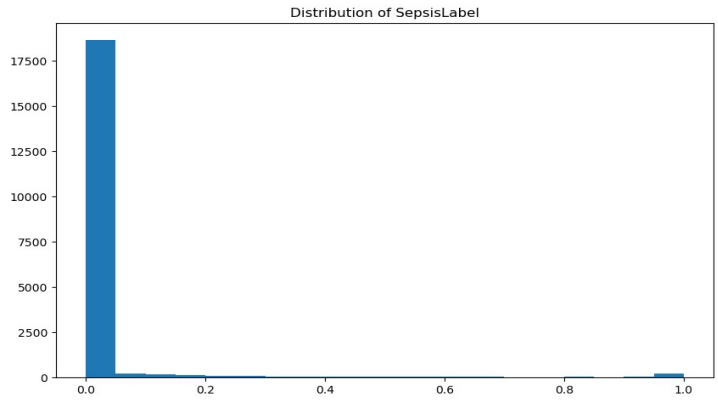


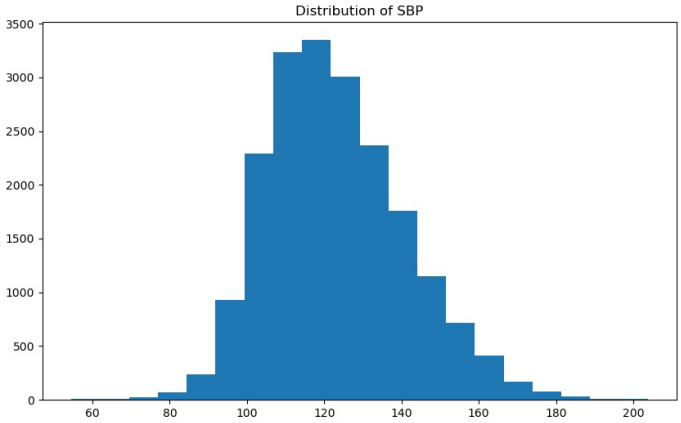


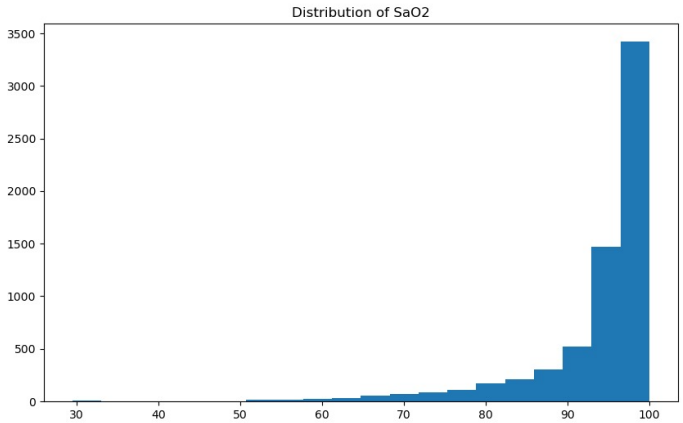


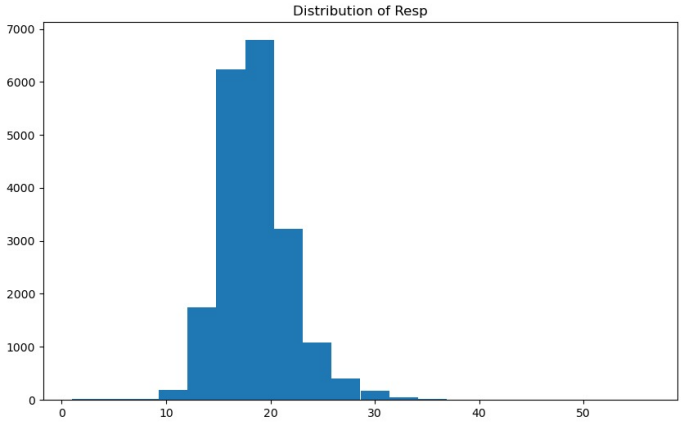


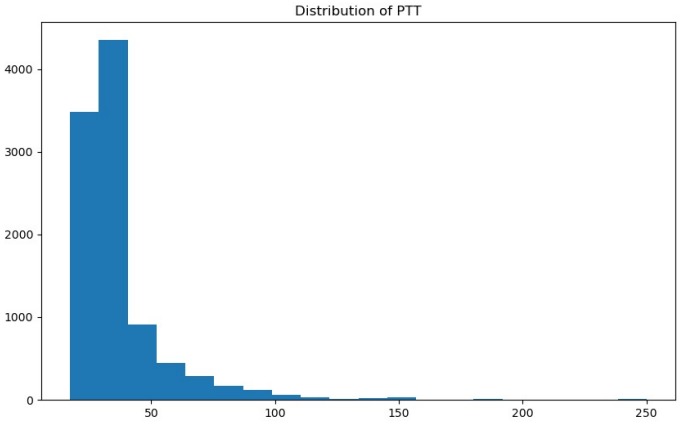


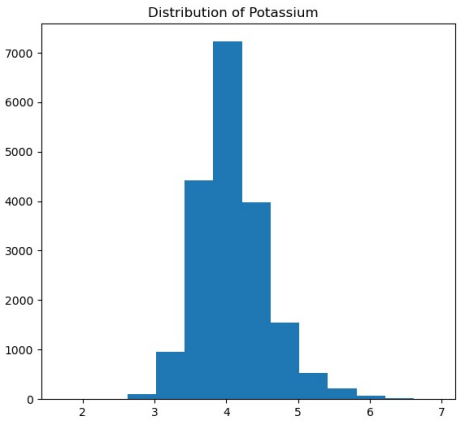


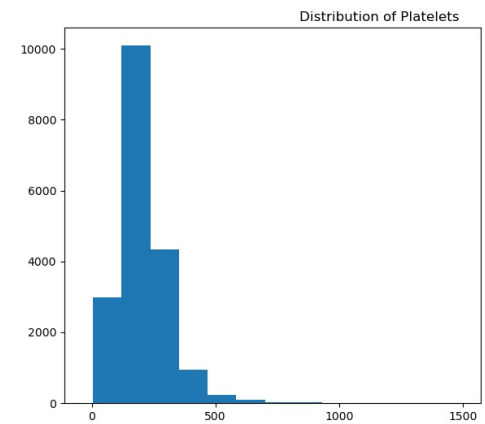

Distribution of SBP
3500
3000
2500
2000
1500
1000
500
0
60
80
100
120
140
160
180
200

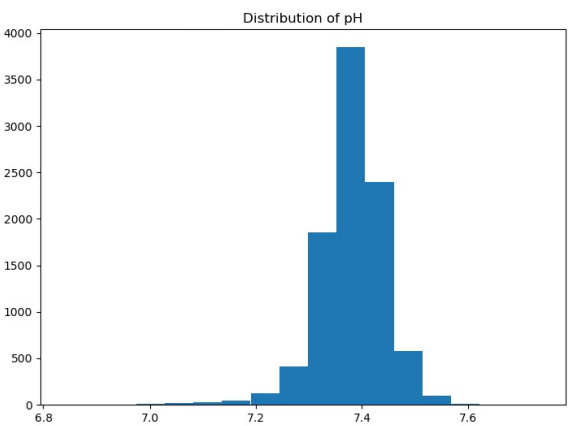

Distribution of SaO2
3500
3000
2500
2000
1500
1000
500
0
30
40
50
60
70
80
90
100

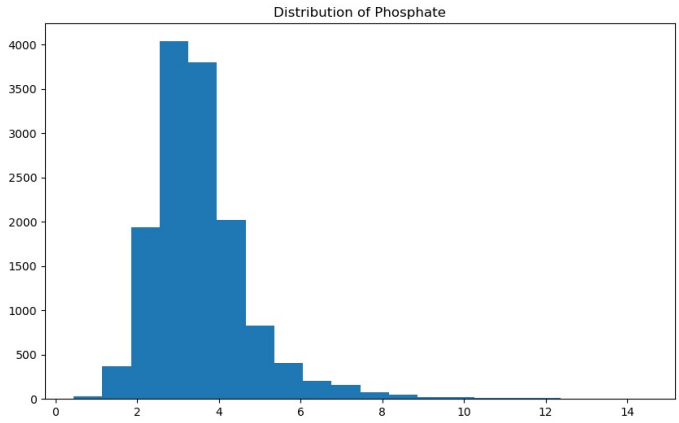

Distribution of Resp
7000
6000
5000
4000
3000
2000
1000
0
0
10
20
30
40
50

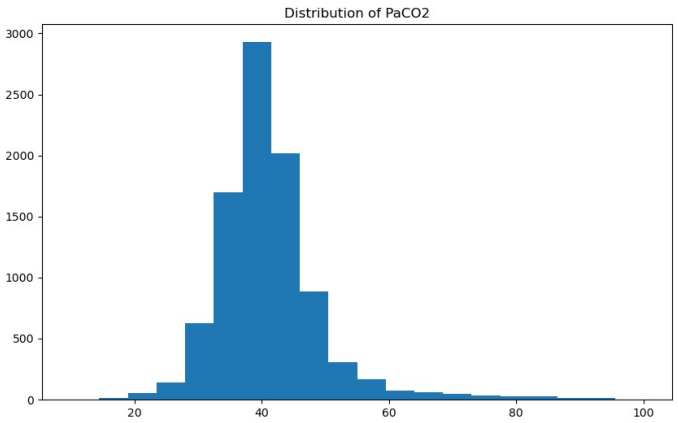

Distribution of PTT
4000
3000
2000
1000
0
50
100
150
200
250

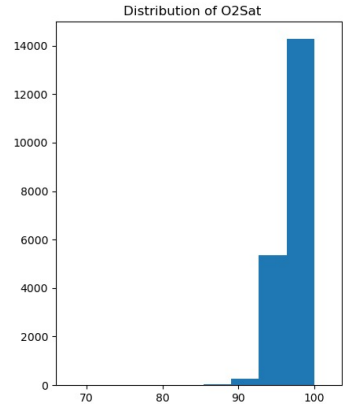

Distribution of Potassium
7000
6000
5000
4000
3000
2000
1000
0
2
3
4
5
6
7

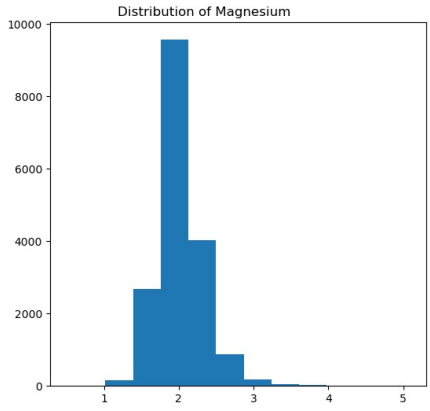

Distribution of Platelets
10000
8000
6000
4000
2000
0
0
500
1000
1500

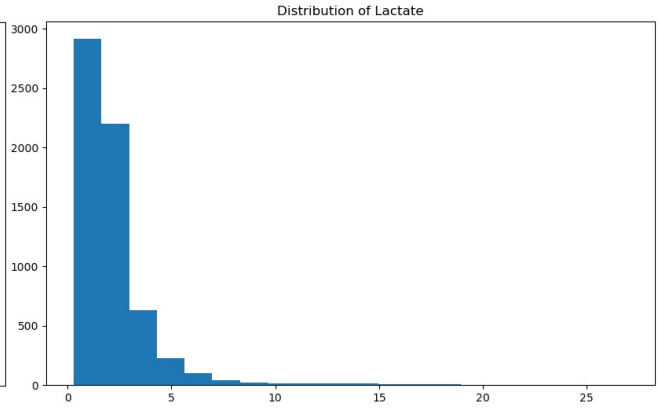

Distribution of pH
4000
3500
3000
2500
2000
1500
1000
500
0
6.8
7.0
7.2
7.4
7.6

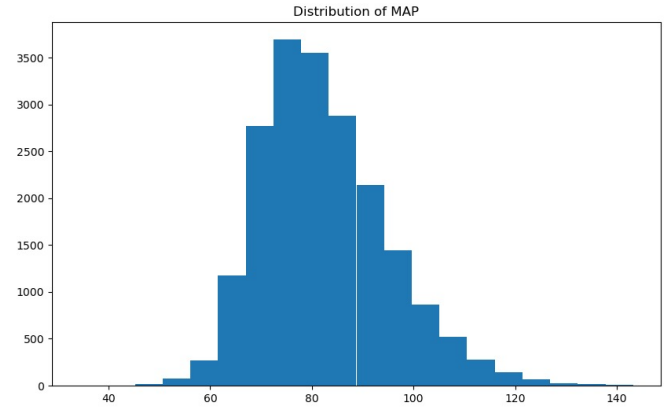

Distribution of Phosphate
4000
3500
3000
2500
2000
1500
1000
500
0
0
2
4
6
8
10
12
14

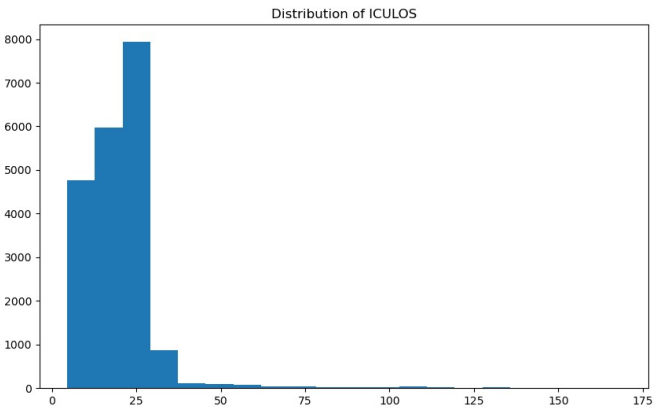

Distribution of PaCO2
3000
2500
2000
1500
1000
500
0
20
40
60
80
100

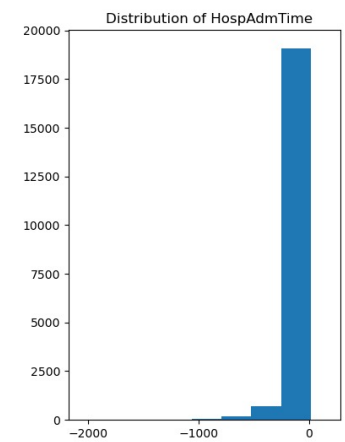

Distribution of O2Sat

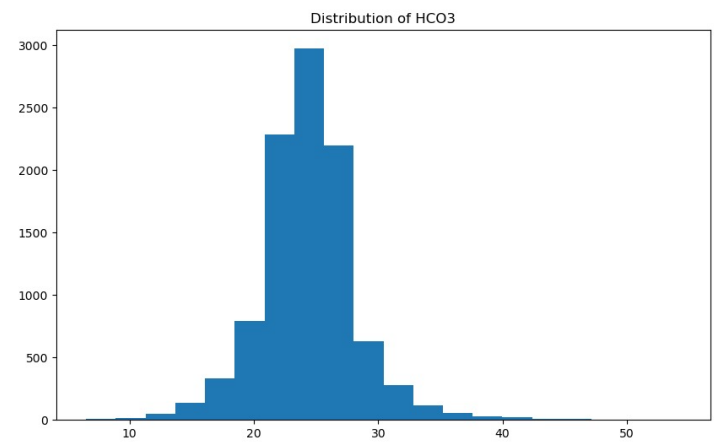

Distribution of Magnesium

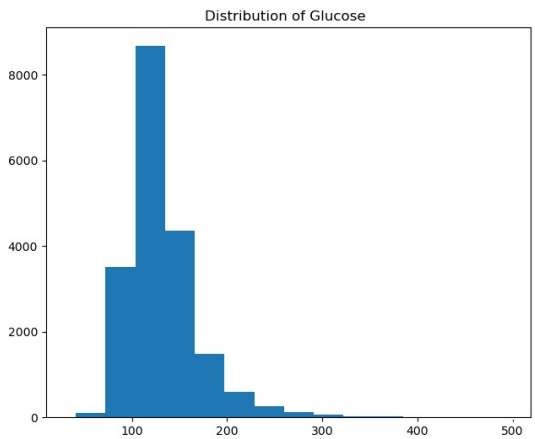

Distribution of Lactate

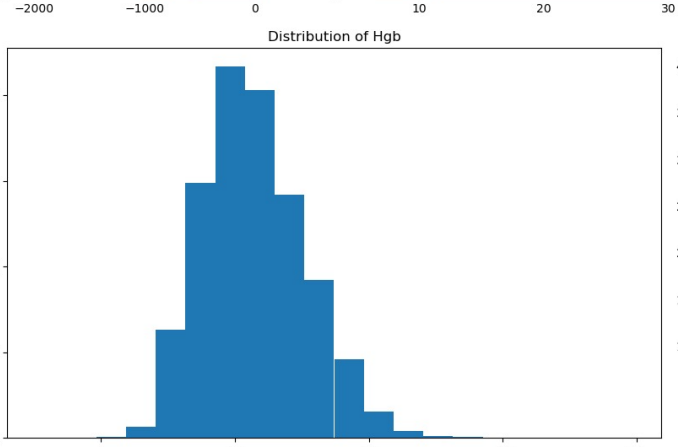

Distribution of MAP

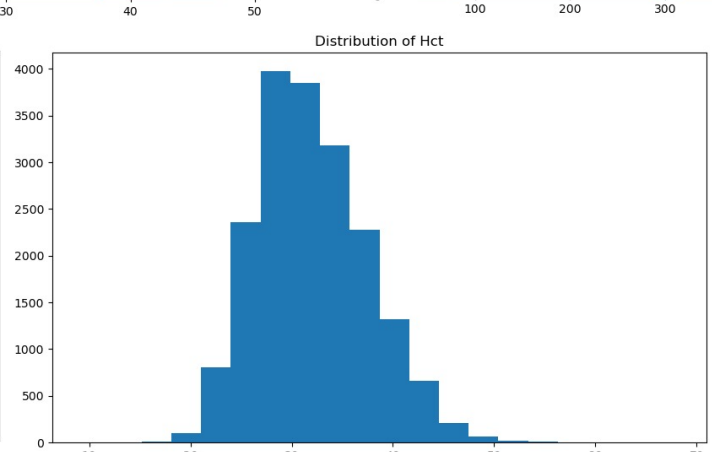

Distribution of ICULOS

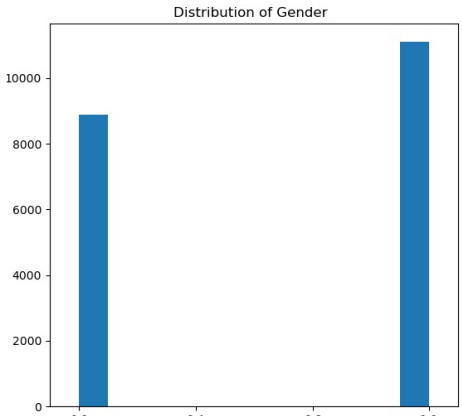

Distribution of HospAdmTime

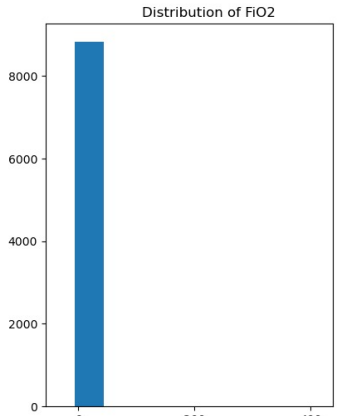

Distribution of HCO3

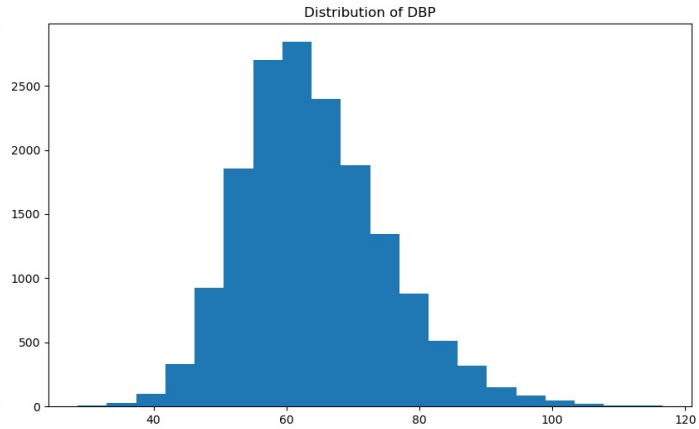

Distribution of Glucose

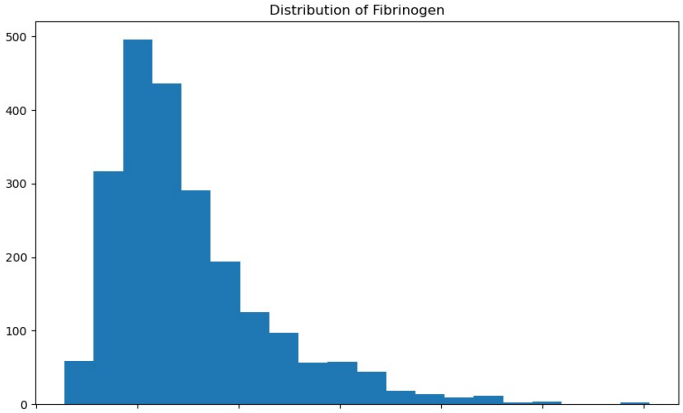

Distribution of Hgb

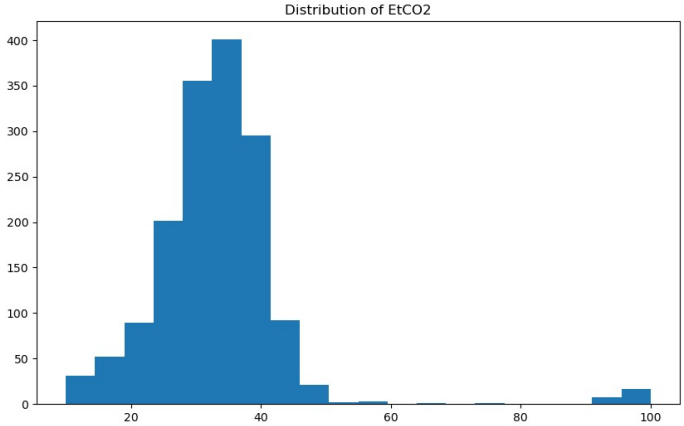

Distribution of Hct

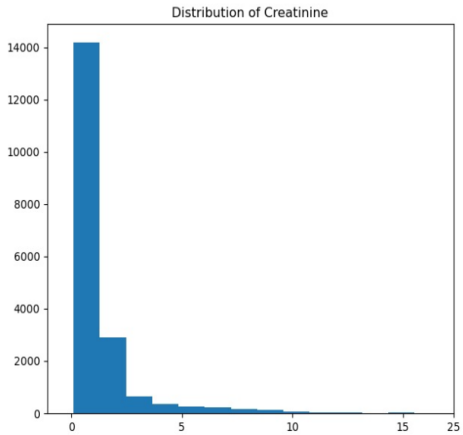

Distribution of Gender

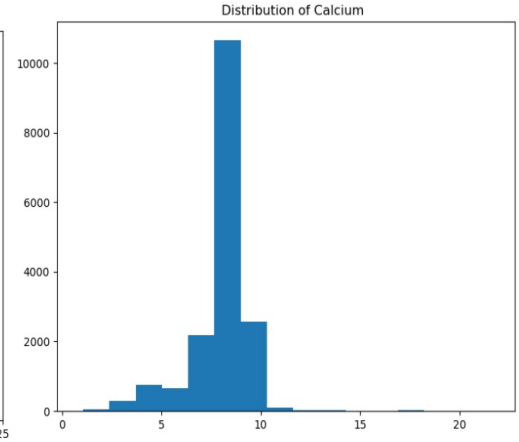

Distribution of FiO2

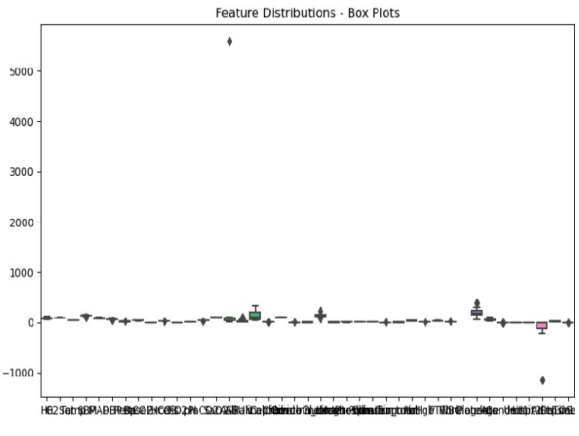

Distribution of DBP

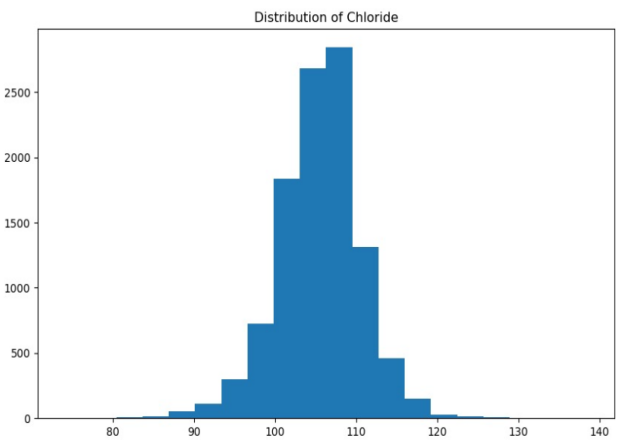

Distribution of Fibrinogen

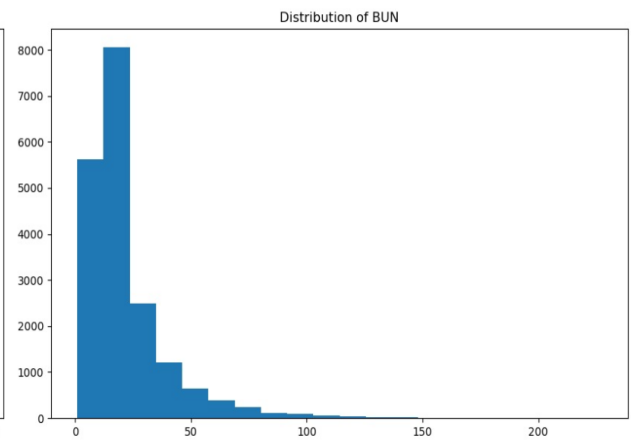

Distribution of EtCO2

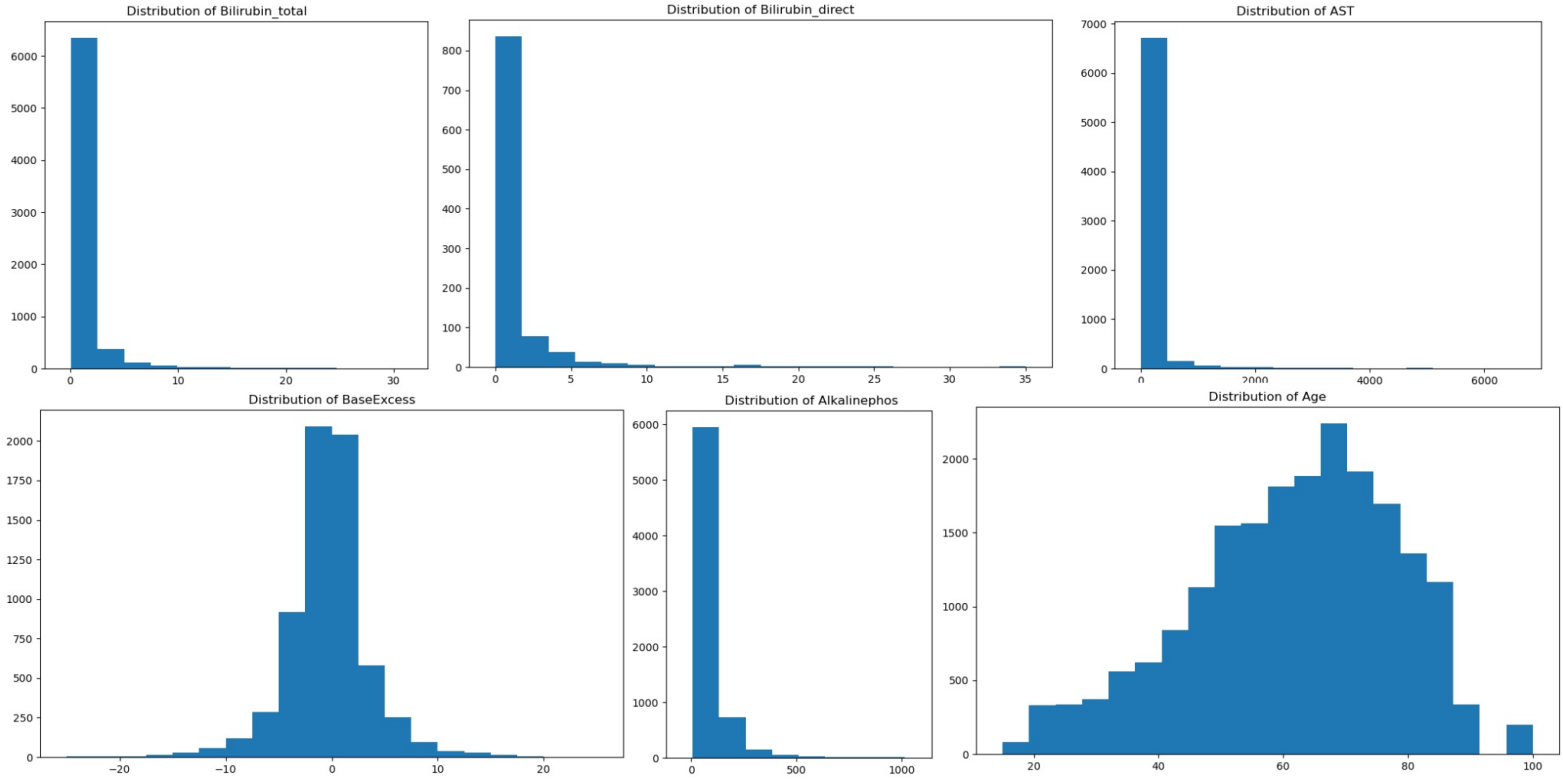

Distribution of Creatinine


Distribution of Calcium


Feature Distributions - Box Plots


Distribution of Chloride


Distribution of BUN



c. Handling missing data:
While working on the preprocess, we had to deal with some missing data (for some patients we had NaN values in some of the features).
On the one hand we wanted to use and extract the most of the data, and on the other hand didn't want to use wrong data.
In order to fulfill our desires, we decided to implement the following protocol:

1. For each patient we checked if there are some features for which all the values are NaN (i.e. the patient has no records with data on the feature), if so, we add the feature into the list we kept.
2. After we iterated over all the patients, we removed for all the patients the features which appeared in the list.
3. After that, we iterated one more time over the patients, where for each patient we used linear interpolation (function of Pandas) in order to complete the missing data and work with full data (or at least some kind of estimation of it).
4. On top of all that, we used the function "fillna" of Pandas in order to make sure we don't have any missing data in the form of NaN.

After all that, we left with **26 features** for each patient.

3. Feature Engineering

a. After we calculated all the data about the features we decided to build a set of features that we want to drop from the all set of features and remain with only relevant features:
DROP SET: **['Alkalinephos', 'PaCO2', 'Bilirubin_total', 'Troponinl', 'Fibrinogen', 'FiO2', 'EtCO2', 'AST', 'PTT', 'Bilirubin_direct', 'pH', 'BaseExcess', 'Lactate', 'SaO2']**
As we believed that there might be some relations between sequential data and features that the LSTM may catch by changing the dimension of the data and using its dependence in time, we strived to use the largest amount of data we could without injecting wrong data to the model (for which the model will learn wrong relations that will lead into bad representation of the data and therefore poor predictions).
Hence, as we stated in section 2.c of the report we used all the features we could use via the constraints we discussed about.

b. We wanted to represent our sequential data in some static representation (vector with some constant dimension) in order to use methods of classification that do not assume that the input is dependent on time (such as SVM, Gradient Boosting classifier).
Therefore, after we preprocessed the data in a way that all the patients have the same features, we pushed each of the patients into one-directional LSTM with one layer with input size of 26 and output size of 100 (as each patient is actually sequential data) and used one of each layers as embedding of the data.
Thanks to the features transformation, we transform the sequential data into the representation of static data which we can use as input for the classifiers (the SVM and the GB).

c. Although we didn't use any enrichment of the data in terms of creating new data in order to get larger dataset, we indeed believe that the change in the dimension of the data added some hidden features.

4. Predictions

After the feature engineering process, we experimented with a few different models. We used the train data to train our models and used the test data to calculate the scores:

1. **Soft-SVM**:

We used Soft-SVM as our first model, we wanted to evaluate the performance of a linear model on the dataset. We used a couple of different kernels and the best kernel that achieved maximum score was RBF Kernel. Also we used

hyperparameter C = 0.5. We also wanted to Calculate also precision, recall and accuracy scores in addition to f1 scores for better comparison. The model achieved this scores:

RBF_SVM Train	**RBF_SVM Test**
Accuracy: 0.96	Accuracy: 0.95
F1 Score: 0.60	**F1 Score: 0.60**
Precision: 0.88	Precision: 0.87
Recall: 0.46	Recall: 0.46

We saw that the model's performance reached the required minimum of F1 score, so we infer that our feature engineering process was successful.

2. **Gradient Boosting Algorithm**:

After testing the SVM model we wanted a model that can handle the imbalanced nature of the data, we wanted to try a different model that is less sensitive to imbalanced data and has regularizations and optimizations. The model hyperparameters:

loss:deviance, learning_rate: 0.1, n_estimators: 100, subsample: 1.0 , min_samples_split: 2, min_samples_leaf: 1, max_depth: 3.

We also wanted to Calculate also precision, recall and accuracy scores in addition to f1 scores for better comparison. The model achieved this scores:

Gradient Boosting Classifier Train	**Gradient Boosting Classifier Test**
Accuracy: 0.97	Accuracy: 0.96
F1 Score: 0.72	**F1 Score: 0.67**
Precision: 0.88	Precision: 0.81
Recall: 0.61	Recall: 0.57

3.**XG BOOST**

After using Gradient Boosting and having better results, we decided to employ XGBoost (eXtreme Gradient Boosting) for several reasons. XGBoost is an advanced implementation of the gradient boosting algorithm, designed to provide better performance and computational efficiency.

We also wanted to Calculate also precision, recall and accuracy scores in addition to f1 scores for better comparisonThe model hyperparameters:

Booster:gbtree , learning_rate: 0.3, n_estimators: 100, max_depth: 6, min_child_weight: 1, gamma: 0 , subsample: 1

We also wanted to Calculate also precision, recall and accuracy scores in addition to f1 scores for better comparison.The model achieved this scores:

XGB Train	**XGB Test**
Accuracy: 0.99	Accuracy: 0.96
F1 Score: 0.91	**F1 Score: 0.68**
Precision: 1.00	Precision: 0.87
Recall: 0.84	Recall: 0.56

d. **Multi-Layer Perceptron (MLP)**:

After the Gradient Boosting Algorithm and SVM results, We decided to explore a different approach and use a stronger model: Multi-Layer Perceptron. The Multi-Layer Perceptron is one of the most powerful machine learning models, utilizing the strengths of deep learning and artificial neural networks.

Our hyperparameters for the model is: Epochs:10 , batch_size: 1, learning_rate: 0.001, input_size: 26, hidden_size:100, output_size: 1, train_size: 0.8, val_size: 0.2,

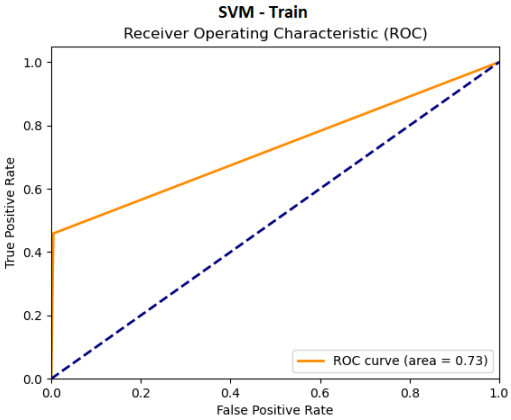
We chose the specific hyperparameters after conducting several trials to identify the optimal configuration for our model. By experimenting with different hyperparameter values, we aimed to find the combination that would yield the best performance in terms of precision, recall, or F1 score.

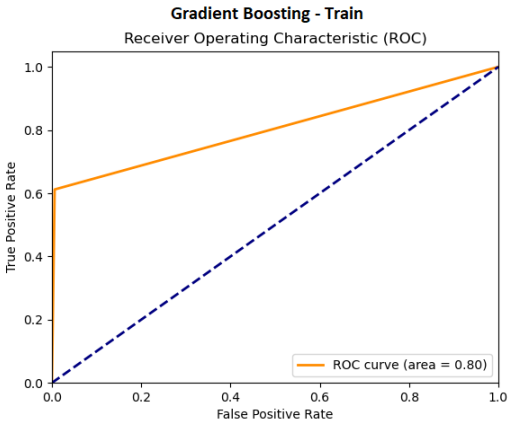
We also wanted to Calculate also precision, recall and accuracy scores in addition to f1 scores for better comparison.The MLP model achieved.

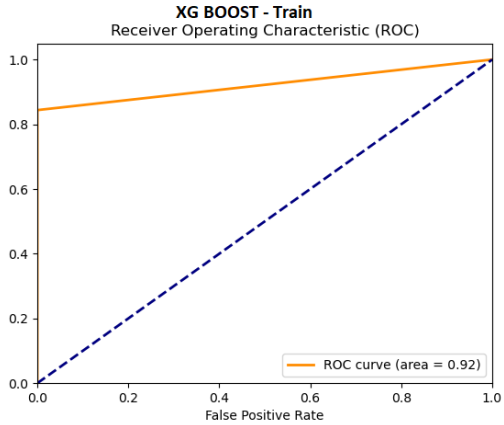
MLP Train	**MLP Test**
Accuracy: 0.95	Accuracy: 0.95
F1 Score: 0.64	**F1 Score: 0.62**
Precision: 0.74	Precision: 0.72
Recall: 0.57	Recall: 0.55

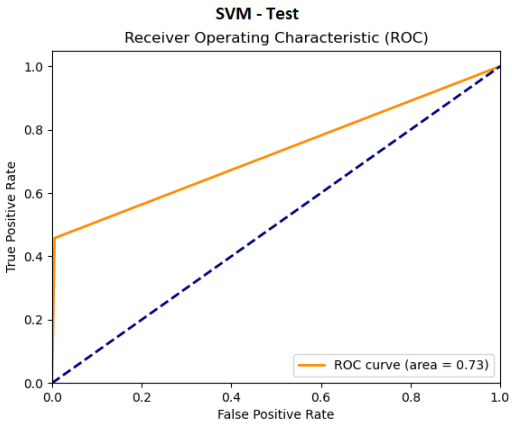
Post Analysis:

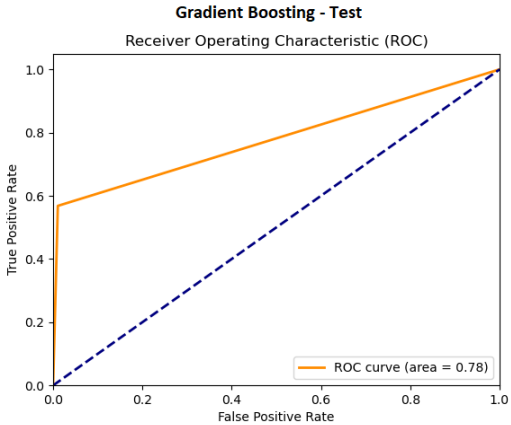
We see that the **XG BOOST** model is the best model that achieved an F1 score of 0.68 on the test and 0.91 score on the train. This is the highest score of all models that we tested. We also drew a Roc - Auc graph for each model for test and train sets:

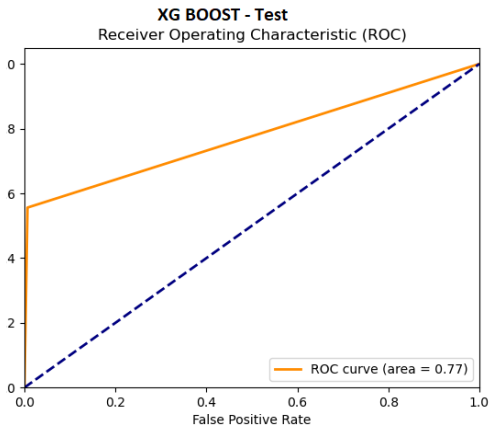


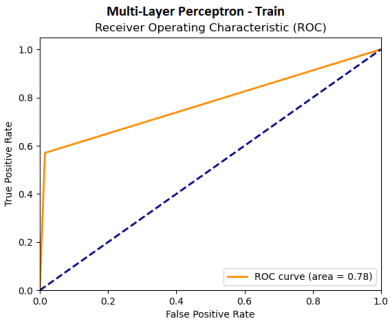


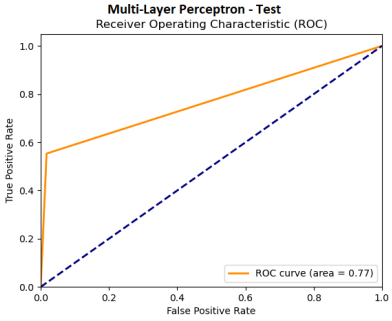












We can see that in all models the Roc - Auc graph on the train set is equal or better than on the test set.

We can see that the XG BOOST model did the best on the train set then other models we tested because he the Closest Auc value to 1. The blue line represent Auc-Roc of a Random model classifier and we can see that all our models

Are much better then the random classifier.

5. Summary and discussion

In conclusion, In this lab we successfully developed and compared various predictive models for sepsis detection using sequential patient data. By applying feature engineering techniques, such as dropping irrelevant columns and extracting meaningful features from the time-series data. We used the LSTM model for extracting the relevant data to a single vector with one label. We streamlined the dataset and improved the models' performance by selecting hyperparameters.

The comparative analysis showed that tree-based algorithms, such as XGBoost, outperformed other models, achieving the highest F1 score of 0.68 on the test. This was expected, as tree-based models are known to handle imbalanced data and high-variance numerical variables effectively. However The Multi-Layer Perceptron (MLP) followed with an F1 score of 0.625.

Altogether, the lab demonstrated the importance of data representation, feature selection, and the choice of algorithms in developing an effective predictive model for sepsis detection. Future work may involve refining the feature engineering process, incorporating additional data sources, exploring neural networks as potential models, and employing ensemble methods with various combinations of models to further enhance the predictive accuracy and utility of the model in real-world applications.