Machine Learning Final Project: Sepsis Prediction

Team Members:

Akshay Jambhulkar(23PGAI004)

Avinash Gupta(23PGAI0085)

Syed Nizamuddin(23PGAI0060)

Vishnuvardhan C(23PGAI0100)

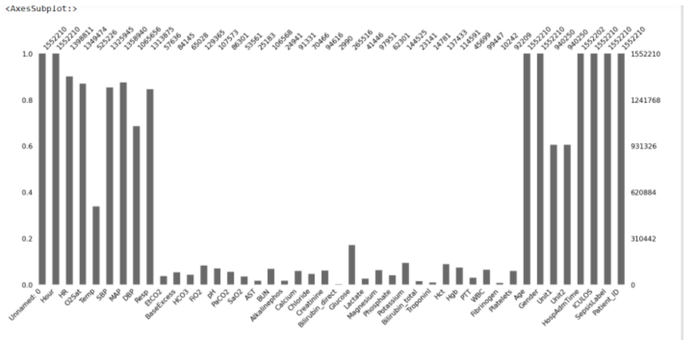
Introduction

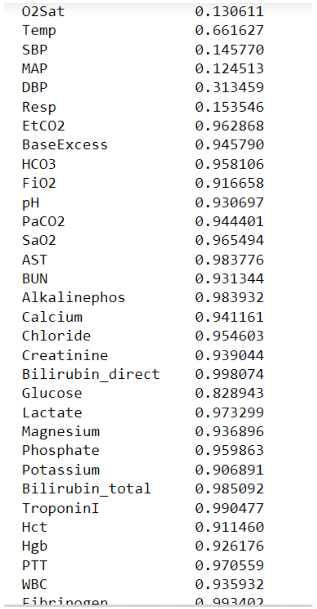
Objective:

The main goal of this project is to use Machine learning algorithms (Supervised & Unsupervised) to predict sepsis diagnosis 6 hours prior to sepsis manifestation using training SetA, training SetB. These datasets are hospital 1 & hospital 2.

Data Preprocessing:

On data preprocessing we find out that many features have missing values, & more than 90% of values are missing for multiple columns in the dataset.





```
O2Sat             0.130611
Temp              0.661627
SBP               0.145770
MAP               0.124513
DBP               0.313459
Resp              0.153546
EtCO2             0.962868
BaseExcess        0.945790
HCO3              0.958106
FiO2              0.916658
pH                0.930697
PaCO2             0.944401
SaO2              0.965494
AST               0.983776
BUN               0.931344
Alkalinephos      0.983932
Calcium           0.941161
Chloride          0.954603
Creatinine        0.939044
Bilirubin_direct  0.998074
Glucose           0.828943
Lactate           0.973299
Magnesium         0.936896
Phosphate         0.959863
Potassium         0.906891
Bilirubin_total   0.985092
TroponinI         0.990477
Hct               0.911460
Hgb               0.926176
PTT               0.970559
WBC               0.935932
Fibrinogen        0.993402
```

Missing values percentage

Introduction

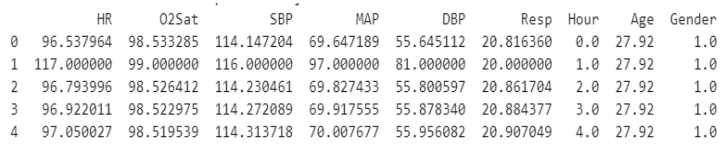
Objective:

The main goal of this project is to use Machine learning algorithms (Supervised & Unsupervised) to predict sepsis diagnosis 6 hrs prior to sepsis manifestation using training SetA, training SetB. These datasets are hospital 1 & hospital 2.

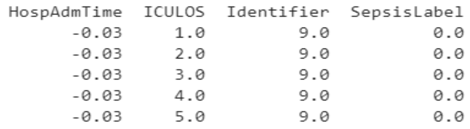
Data Set:

- We are using physiological data which is hosted on PhysioNet Computing in Cardiology Challenge 2019.
- Retrieving the dataset which is having pipe separated features for respective timestamp.
- Data contains 40 feature consisting of:
  - 8 Vitals Signs – Heart Rate, Temperature, MAP, …
  - 28 Laboratory Values – FiO2, Lactate, Bilirubin, …
  - 6 Demographics – Age, Gender, Hospital Unit, …

Example of one patient dataset post imputation:



```
           HR      O2Sat         SBP        MAP        DBP       Resp  Hour    Age  Gender
0   96.537964  98.533285  114.147204  69.647189  55.645112  20.816360   0.0  27.92     1.0
1  117.000000  99.000000  116.000000  97.000000  81.000000  20.000000   1.0  27.92     1.0
2   96.793996  98.526412  114.230461  69.827433  55.800597  20.861704   2.0  27.92     1.0
3   96.922011  98.522975  114.272089  69.917555  55.878340  20.884377   3.0  27.92     1.0
4   97.050027  98.519539  114.313718  70.007677  55.956082  20.907049   4.0  27.92     1.0
```



```
HospAdmTime  ICULOS  Identifier  SepsisLabel
      -0.03     1.0         9.0          0.0
      -0.03     2.0         9.0          0.0
      -0.03     3.0         9.0          0.0
      -0.03     4.0         9.0          0.0
      -0.03     5.0         9.0          0.0
```

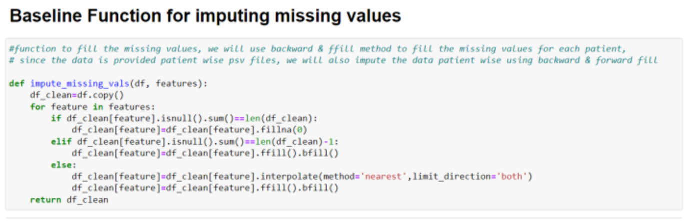
Data Preprocessing:

- On data preprocessing we find out that many features have missing values, & more than 90% of values are missing for multiple columns in the dataset.

Data imputation & preparing training , test & validation files

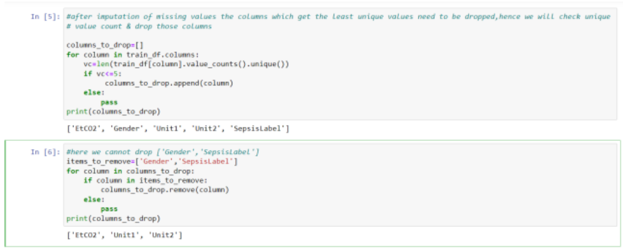
Data imputation & Data split into train, test & validation:

- Linear imputation (forward & backward fill) with nearest neighbor.
- Imputed each patient separately before merging the .psv files as a single file.
- Patients (.psv files) in Training Set A was divided in three different sets: Training – 10%, Validation Set -15% and Test- 15%



Baseline Function for imputing missing values

```
#function to fill the missing values, we will use backward & ffill method to fill the missing values for each patient,
# since the data is provided patient wise psv files, we will also impute the data patient wise using backward & forward fill

def impute_missing_vals(df, features):
    df_clean=df.copy()
    for feature in features:
        if df_clean[feature].isnull().sum()==len(df_clean):
            df_clean[feature]=df_clean[feature].fillna(0)
        elif df_clean[feature].isnull().sum()==len(df_clean)-1:
            df_clean[feature]=df_clean[feature].ffill().bfill()
        else:
            df_clean[feature]=df_clean[feature].interpolate(method='nearest',limit_direction='both')
            df_clean[feature]=df_clean[feature].ffill().bfill()
    return df_clean
```



```
In [5]: #after imputation of missing values the columns which get the least unique values need to be dropped,hence we will check unique
        # value count & drop those columns

        columns_to_drop=[]
        for column in train_df.columns:
            vc=len(train_df[column].value_counts().unique())
            if vc<=5:
                columns_to_drop.append(column)
            else:
                pass
        print(columns_to_drop)

        ['EtCO2', 'Gender', 'Unit1', 'Unit2', 'SepsisLabel']

In [6]: #here we cannot drop ['Gender','SepsisLabel']
        items_to_remove=['Gender','SepsisLabel']
        for column in columns_to_drop:
            if column in items_to_remove:
                columns_to_drop.remove(column)
            else:
                pass
        print(columns_to_drop)

        ['EtCO2', 'Unit1', 'Unit2']
```

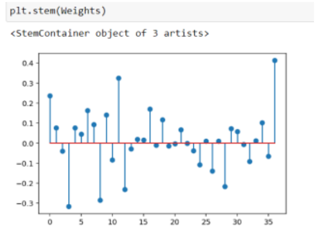
Dropped above features

- Above same procedure was applied for Training Set B.
- After data imputation we looked at the features with continuous data which have unique values less than 5 & drop these features. These features were: EtCO2, Unit1, Unit2

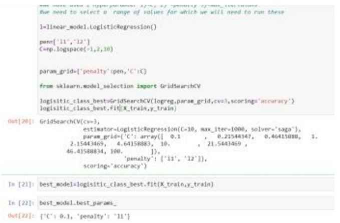
Model Application-Logistic Regression

Logistic Regression for Classification-1:

- Logistic Regression Classifier was applied & the weight coefficients of individual features were interpreted. The 4 most important features were observed to be [ICULOS,PaCO2,SBP,HCO3]. These 4 features contribute most for the prediction of Sepsis disease.



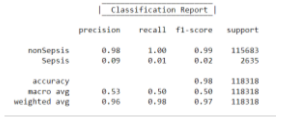
```
plt.stem(Weights)

<StemContainer object of 3 artists>
```



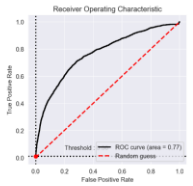
```
[illegible]
#we need to select a  range of values for which we will need to run these

l=linear_model.LogisticRegression()

pen=['l1','l2']
C=np.logspace(-1,2,10)


param_grid={'penalty':pen,'C':C}

from sklearn.model_selection import GridSearchCV

logisitic_class_best=GridSearchCV(logreg,param_grid,cv=3,scoring='accuracy')
logisitic_class_best.fit(X_train,y_train)

Out[20]: GridSearchCV(cv=3,
             estimator=LogisticRegression(C=10, max_iter=1000, solver='saga'),
             param_grid={'C': array([  0.1       ,   0.21544347,   0.46415888,   1.
         2.15443469,   4.64158883,  10.        ,  21.5443469 ,
        46.41588834, 100.        ]),
                         'penalty': ['l1', 'l2']},
             scoring='accuracy')

In [21]: best_model=logisitic_class_best.fit(X_train,y_train)

In [22]: best_model.best_params_

Out[22]: {'C': 0.1, 'penalty': 'l1'}
```

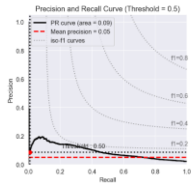
Hyperparametric Tuning & Performance Metrics:

Hyperparametric tuning was done & the best parameters were found.   i.e {'C':0.1, 'penalty': 'l1'} . The model was run using the obtained hyper-parameters. The performance of the model was evaluated based on precision, recall, f1 score and roc curve area.



```
                    | Classification Report |

              precision    recall  f1-score   support

   nonSepsis       0.98      1.00      0.99    115683
      Sepsis       0.09      0.01      0.02      2635

    accuracy                           0.98    118318
   macro avg       0.53      0.50      0.50    118318
weighted avg       0.96      0.98      0.97    118318
```



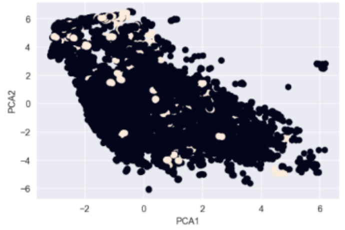


Observations: The ROC Curve (Area) for the Logistic Regression model is 0.77, Precision is 9%.

Model Application 2- PCA Along with Logistic Regression

Logistic Regression for Classification along with PCA:

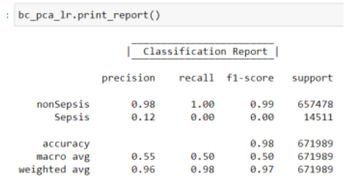
PCA was applied to the pre-processed data, and the scatter plot for 1st and 2nd principal component was plotted. Post PCA, hyper parameter tuning was done to find the optimal number of principal components which gives the max information about the original features. Number of principal components to be selected were found out to be 15.



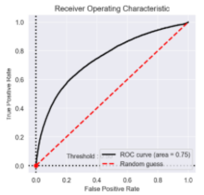
PCA 1 vs PCA 2

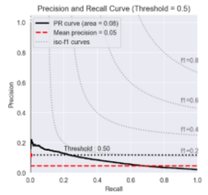
Performance metrics:

Hyperparametric tuning was done & the best parameters were found. i.e {'n_components=15}. The model was run using the obtained hyper-parameters. The performance of the model was evaluated based on precision, recall, f1 score and roc curve area.



```
bc_pca_lr.print_report()

                 Classification Report

              precision    recall  f1-score   support

   nonSepsis       0.98      1.00      0.99    657478
      Sepsis       0.12      0.00      0.00     14511

    accuracy                           0.98    671989
   macro avg       0.55      0.50      0.50    671989
weighted avg       0.96      0.98      0.97    671989
```



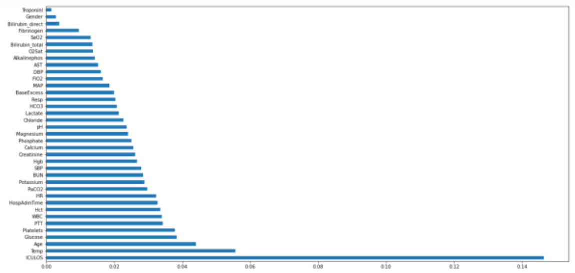


Observations: The ROC Curve (Area) for the PCA Logistic Regression model is 0.75, Precision is 12%.

Model Application 3- Random Forest

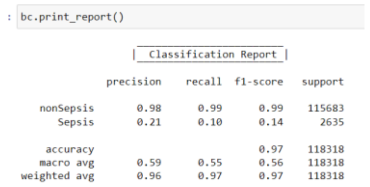
Random Forest for Classification:

Ensemble learning model (Random Forest) was applied to the dataset. The feature importance were found out and plotted as below**.** The feature importance was decided based on the features capability to detect sepsis**.**

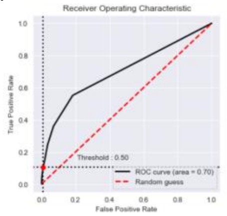


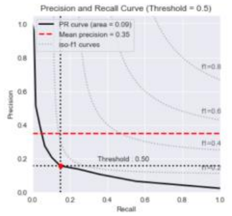
Performance metrics :

The model was run and the performance of the model was evaluated based on precision, recall, f1 score and roc curve area.



```
: bc.print_report()

                    | Classification Report |

              precision    recall  f1-score   support

   nonSepsis       0.98      0.99      0.99    115683
      Sepsis       0.21      0.10      0.14      2635

    accuracy                           0.97    118318
   macro avg       0.59      0.55      0.56    118318
weighted avg       0.96      0.97      0.97    118318
```





Observations: The ROC Curve (Area) for the Random Forest model is 0.70, Precision is 21%.

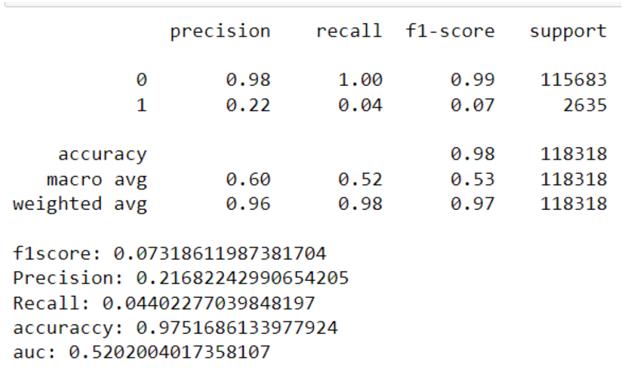
Model Application 4- xgboost

Xgboost for Classification:

Another ensembling model XGBoost was applied to the dataset.

Performance metrics :

The model was run and the performance of the model was evaluated based on precision, recall, f1 score and roc curve area

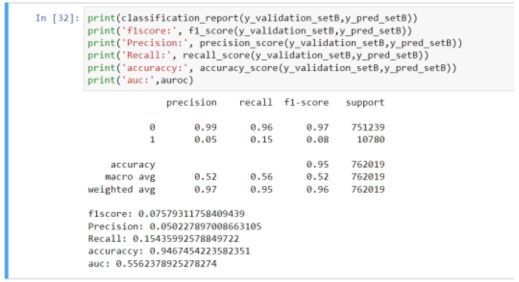


```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99    115683
           1       0.22      0.04      0.07      2635

    accuracy                           0.98    118318
   macro avg       0.60      0.52      0.53    118318
weighted avg       0.96      0.98      0.97    118318

f1score: 0.07318611987381704
Precision: 0.21682242990654205
Recall: 0.04402277039848197
accuraccy: 0.9751686133977924
auc: 0.5202004017358107
```

Observations: The ROC Curve (Area) for the xgboost model is 0.52, Precision is 22%.

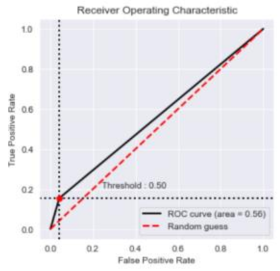
Results:

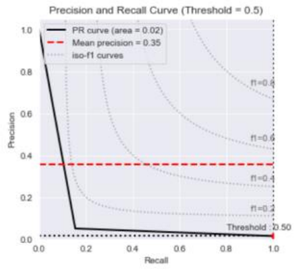
Results: After analyzing the 4 models , we are proceeding with Random Forest Classifier for the final deployment model & will test the Random Forest Classifier on the TrainingSetB (i.e. validation set)



```
In [32]: print(classification_report(y_validation_setB,y_pred_setB))
         print('f1score:', f1_score(y_validation_setB,y_pred_setB))
         print('Precision:', precision_score(y_validation_setB,y_pred_setB))
         print('Recall:', recall_score(y_validation_setB,y_pred_setB))
         print('accuraccy:', accuracy_score(y_validation_setB,y_pred_setB))
         print('auc:',auroc)
```

```
              precision    recall  f1-score   support

           0       0.99      0.96      0.97    751239
           1       0.05      0.15      0.08     10780

    accuracy                           0.95    762019
   macro avg       0.52      0.56      0.52    762019
weighted avg       0.97      0.95      0.96    762019

f1score: 0.07579311758409439
Precision: 0.050227897008663105
Recall: 0.15435992578849722
accuraccy: 0.9467454223582351
auc: 0.5562378925278274
```





Observations: The ROC Curve (Area) for the Random Forest model is 0.52, Precision is 22%.

Conclusion:

Random Forest has highest diagnostic accuracy among ensemble method.

Thanks