Data Mining

SI 515

Topic:

Classification on Parkinsons Diseased based on ML algorithms

Submitted by:

Tushar Khatri (22N0066)

Supervised by:

Prof. Radhendushka Srivastava and Prof. Sanjeev V Sabnis

Department of Mathematics | IIT Bombay

Acknowledgement

I would like to thank my project supervisor **Prof. Radhendushka Srivastava and Prof. Sanjeev V Sabnis** for their unwavering support and expert guidance throughout this project. Theiradvice, feedback, and constructive criticism have been instrumental in shaping the direction of this project and improving its quality. I am truly grateful for their generosity with their time and expertise.

Furthermore, I would like to express my immense gratitude to everyone whohas supported me throughout this project. Your encouragement, guidance, and assistance have been invaluable and have enabled me to complete this project successfully

Lastly, I would like to thank all the resources and references that have been instrumental in shaping the ideas and concepts presented in this project. Theknowledge and insights gained from these sources have been invaluable andhave contributed significantly to the success of this project.

Thank you all once again for your support, guidance, and assistance. This project would not have been possible without you all. I would like to expressmy immense gratitude to everyone who has supported me throughout this project. Your encouragement, guidance, and assistance have been invaluableand have enabled me to complete this project successfully

Tushar Khatri (22N0066) Date: 04/11/2023

Chapter-1

Objective

The objective of this classification project using the Parkinson's dataset is to develop a model that can classify the disease (Yes / No) of a person based on its various characteristics, such as its vocal frequency, signal fractal scaling experiment and many others. This dataset is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). Each column in the table is a particular voice measure, and each row corresponds to one of 195 voice recordings from these individuals ("name" column). The main aim of the data is to discriminate healthy people from those with PD, according to the "status" column which is set to 0 for healthy and 1 for PD.

The goal is to develop a ML model that accurately classify the new observation based on their characteristics, with the ultimate objective of helping doctors and patients to make more informed decisions about medical conditions

The success of the project is measured by the model's ability to accurately classify status value (Yes/No) for disease based on new data that was not used during the training process**.** Finally at the end we obtain all necessary results, conclusions alongwith interpretation.

Chapter-2

Data Description

Data Set Information:

The data used in this study were gathered from patients with PD with ages ranging from 33 to 87 at the Department of Neurology in Faculty of Medicine, Istanbul University. The control group consists of some healthy individuals with ages varying between 41 and 82. During the data collection process, the microphone is set to 44.1 KHz and following the physician’s examination, the sustained phonation of the vowel was collected from each subject with three repetitions.

Attribute Information:

Various speech signal processing algorithms including Time Frequency Features, Mel Frequency Cepstral Coefficients (MFCCs), Wavelet Transform based Features, Vocal Fold Features and TWQT features have been applied to the speech recordings of Parkinson's Disease (PD) patients to extract clinically useful information for PD assessment.

Dataset Source:
This dataset is collected from UCI Machine Learning Repository through the following link:
https://archive.ics.uci.edu/ml/datasets/Parkinson%27s+Disease+Classification
https://archive.ics.uci.edu/dataset/174/parkinsons

Citation Request:

If you use this dataset, please cite: Sakar, C.O., Serbes, G., Gunduz, A., Tunc, H.C., Nizam, H., Sakar, B.E., Tutuncu, M., Aydin, T., Isenkul, M.E. and Apaydin, H., 2018. A comparative analysis of speech signal processing algorithms for Parkinsonâ€™s disease classification and the use of the tunable Q-factor wavelet transform. Applied Soft Computing, DOI:
[Web Link] **https://doi.org/10.1016/j.asoc.2018.10.022**

Attributes Information:

name - ASCII subject name and recording number

MDVP:Fo(Hz) - Average vocal fundamental frequency

MDVP:Fhi(Hz) - Maximum vocal fundamental frequency

MDVP:Flo(Hz) - Minimum vocal fundamental frequency

MDVP:Jitter(%) , MDVP:Jitter(Abs) , MDVP:RAP , MDVP:PPQ , Jitter:DDP - Several measures of variation in fundamental frequency

MDVP:Shimmer , MDVP:Shimmer(dB) , Shimmer:APQ3 , Shimmer:APQ5 , MDVP:APQ , Shimmer:DDA - Several measures of variation in amplitude

NHR , HNR - Two measures of ratio of noise to tonal components in the voice

status - Health status of the subject (one) - Parkinson's, (zero) - healthy

RPDE , D2 - Two nonlinear dynamical complexity measures

DFA - Signal fractal scaling exponent

spread1 , spread2 , PPE - Three nonlinear measures of fundamental frequency variation

Research Paper:

https://www.sciencedirect.com/science/article/pii/S1877050918319793

What is Parkison's Disease?

Parkinson's is a neurodegenerative progressive disorder disease. It affects the nervous system and the part of the body controlled by it. It appears in the part of the brain called substantia nigra. Normally these brain cells produce dopamine and this dopamine operates in a delicate balance with other neurotransmitters to help coordinate the millions of nerves and muscle cell involved in the movement. When these nerve cells die or become impaired, they lose the ability to produce the dopamine. When 60-80% of these cells are lost then enough dopamine are not produced and Parkinson's motor symptoms appear. Without enough dopamine, this balance is disrupted resulting tremor, rigidity, slowness of movement and impaired balance.

So, Parkinson's disease is caused by the disruption of the brain cells that produce the dopamine which allows the brain to communicate each other and control the fluency of the movement. It is thought that the disease begins many years before the motor symptoms appears and therefore, researchers are looking for ways to recognize the non-motor symptoms that appear early in the disease as early as possible, thereby halting the progression of the disease. A person suffering from the Parkinson's can have following symptoms:

- Tremor
- Slowed movement
- Rigid muscle
- Impaired posture
- Loss of automatic movement
- Speech change
- Writing change
- Impaired posture Also, Parkinson's disease can have other symptoms that include:
- Depression

- Anxiety
- Sleeping and memory-related issues
- Loss of sense of smell along with balance problems

The symptoms of Parkinson's disease can be different from patient to patient. Early sign of the disease can be mild and can go unnoticed. Symptoms often begin on one side of your body and usually remain worse on that side. Sometimes it is difficult to detect whether there is Parkinson's disease present in the patient's body. Parkinson's disease if detected in the early stage will be curable, and will be time and cost effective, but there is no effective treatment in the advanced stage.

Understanding the dataset

Data Set Information:

This dataset is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). Each column in the table is a particular voice measure, and each row corresponds to one of 195 voice recordings from these individuals ("name" column). The main aim of the data is to discriminate healthy people from those with PD, according to the "status" column which is set to 0 for healthy and 1 for PD.

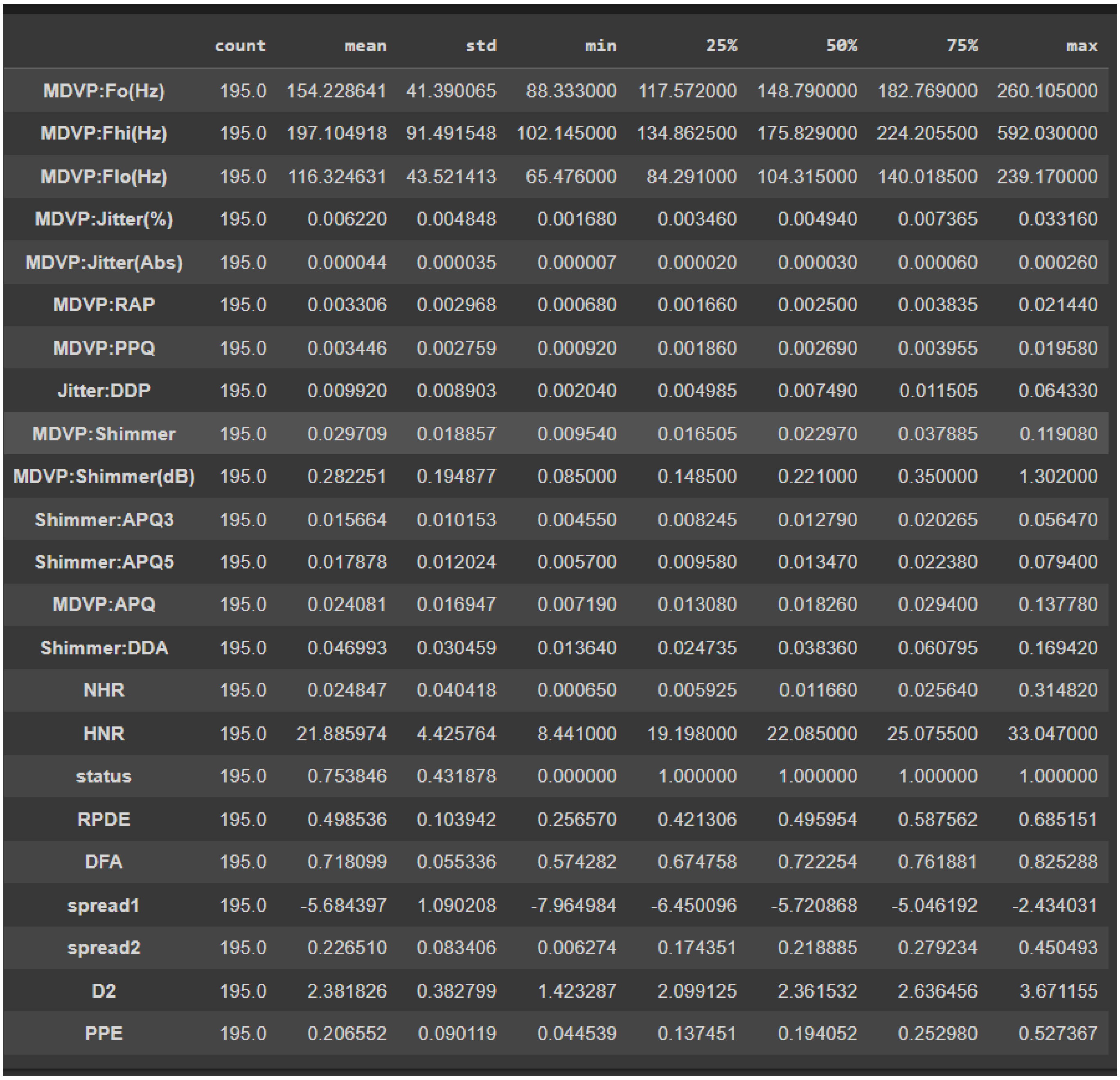
'0' refers to (B)enign

'1' refers to (M)alignant

Chapter – 3

Exploratory Data Analysis

Summary of the data:



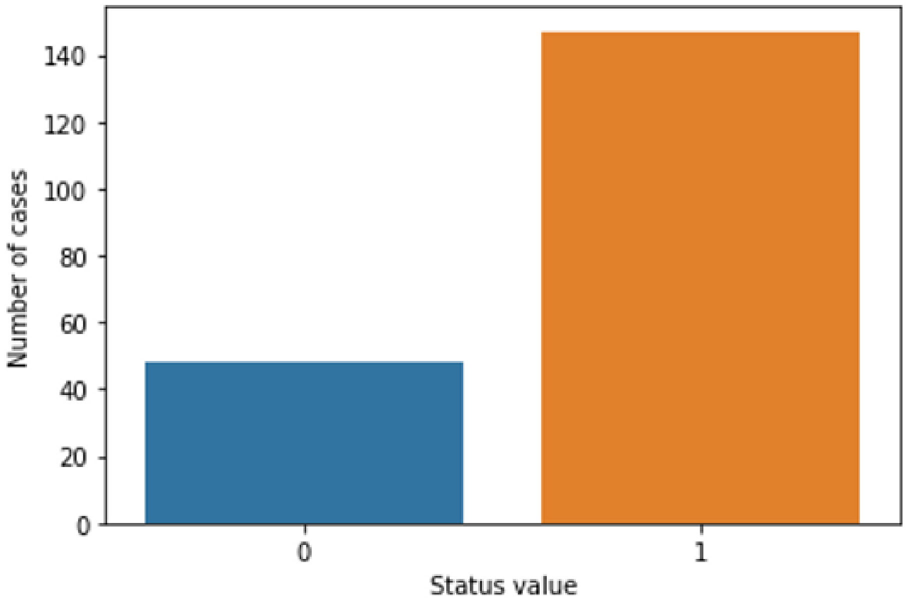
	count	mean	std	min	25%	50%	75%	max
MDVP:Fo(Hz)	195.0	154.228641	41.390065	88.333000	117.572000	148.790000	182.769000	260.105000
MDVP:Fhi(Hz)	195.0	197.104918	91.491548	102.145000	134.862500	175.829000	224.205500	592.030000
MDVP:Flo(Hz)	195.0	116.324631	43.521413	65.476000	84.291000	104.315000	140.018500	239.170000
MDVP:Jitter(%)	195.0	0.006220	0.004848	0.001680	0.003460	0.004940	0.007365	0.033160
MDVP:Jitter(Abs)	195.0	0.000044	0.000035	0.000007	0.000020	0.000030	0.000060	0.000260
MDVP:RAP	195.0	0.003306	0.002968	0.000680	0.001660	0.002500	0.003835	0.021440
MDVP:PPQ	195.0	0.003446	0.002759	0.000920	0.001860	0.002690	0.003955	0.019580
Jitter:DDP	195.0	0.009920	0.008903	0.002040	0.004985	0.007490	0.011505	0.064330
MDVP:Shimmer	195.0	0.029709	0.018857	0.009540	0.016505	0.022970	0.037885	0.119080
MDVP:Shimmer(dB)	195.0	0.282251	0.194877	0.085000	0.148500	0.221000	0.350000	1.302000
Shimmer:APQ3	195.0	0.015664	0.010153	0.004550	0.008245	0.012790	0.020265	0.056470
Shimmer:APQ5	195.0	0.017878	0.012024	0.005700	0.009580	0.013470	0.022380	0.079400
MDVP:APQ	195.0	0.024081	0.016947	0.007190	0.013080	0.018260	0.029400	0.137780
Shimmer:DDA	195.0	0.046993	0.030459	0.013640	0.024735	0.038360	0.060795	0.169420
NHR	195.0	0.024847	0.040418	0.000650	0.005925	0.011660	0.025640	0.314820
HNR	195.0	21.885974	4.425764	8.441000	19.198000	22.085000	25.075500	33.047000
status	195.0	0.753846	0.431878	0.000000	1.000000	1.000000	1.000000	1.000000
RPDE	195.0	0.498536	0.103942	0.256570	0.421306	0.495954	0.587562	0.685151
DFA	195.0	0.718099	0.055336	0.574282	0.674758	0.722254	0.761881	0.825288
spread1	195.0	-5.684397	1.090208	-7.964984	-6.450096	-5.720868	-5.046192	-2.434031
spread2	195.0	0.226510	0.083406	0.006274	0.174351	0.218885	0.279234	0.450493
D2	195.0	2.381826	0.382799	1.423287	2.099125	2.361532	2.636456	3.671155
PPE	195.0	0.206552	0.090119	0.044539	0.137451	0.194052	0.252980	0.527367

Observation:

- From the above summary we can see clearly that mean of features varying a lot in scale also we can see this from std in the above summary table so this suggests we need scaling while applying certain algorithm in future.
- We don’t have null /missing value in our dataset

Univariate Analysis:

Number of Parkinson's Disease patients: 147 (75.38%) Number of Healthy patients: 48 (24.62%)

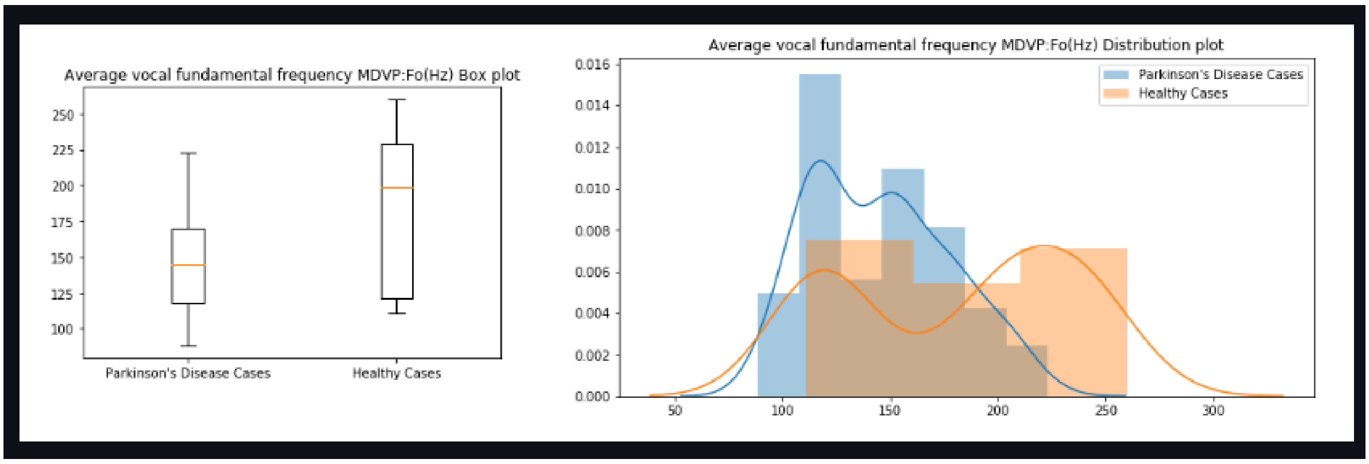


From above we can see clearly data is biased towards no of disease patients but since no of healthy patients to no of disease patients are 25:75 approximately so the data is moderately biased. We will look below how to deal with this issue.

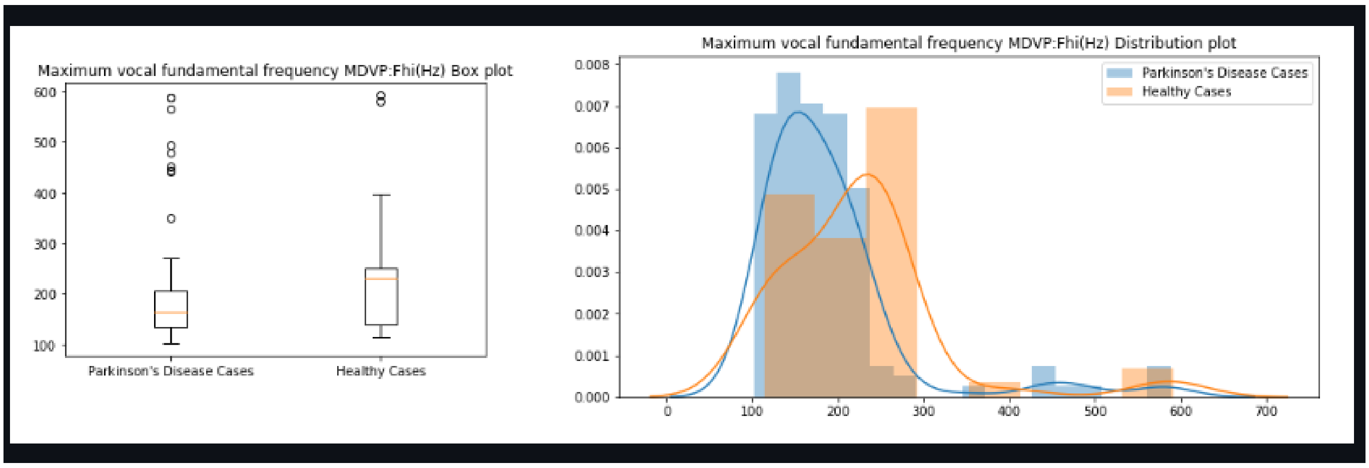
Analysis on fundamental frequencies:

The data set has 195 samples. Each row of the data set consists of voice recording of individuals with name and 23 attributes of biomedical voice measurements. The main aim of the data is to discriminate healthy people from those with Parkinson's Disease, according to "status" column which is set to \`0\` for healthy and \`1\` for individual affected with Parkinson's Disease.

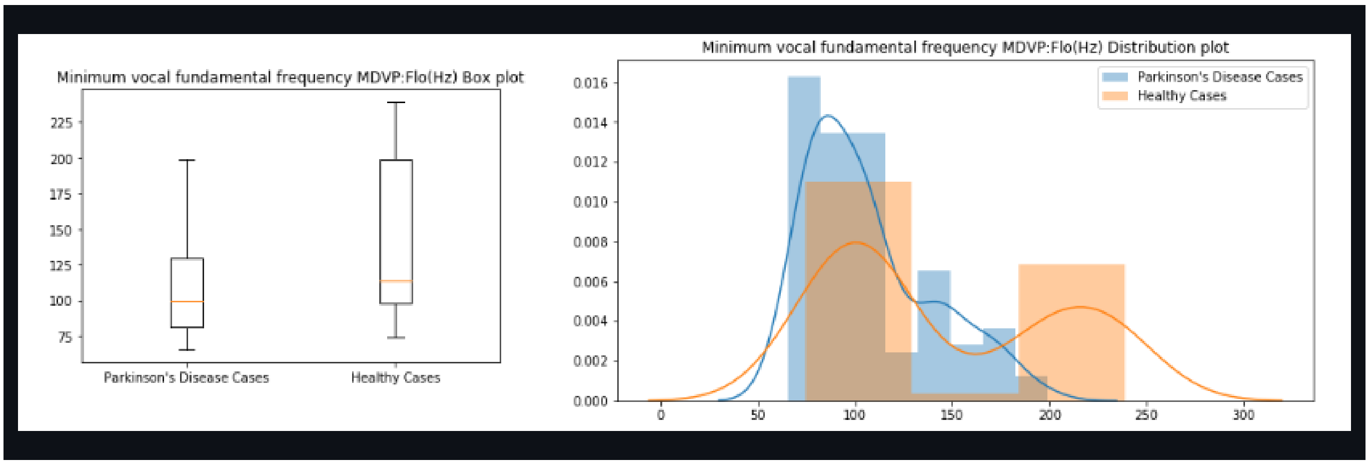
3.1) For Average fundamental frequencies



3.2) For high(Maximum) frequency



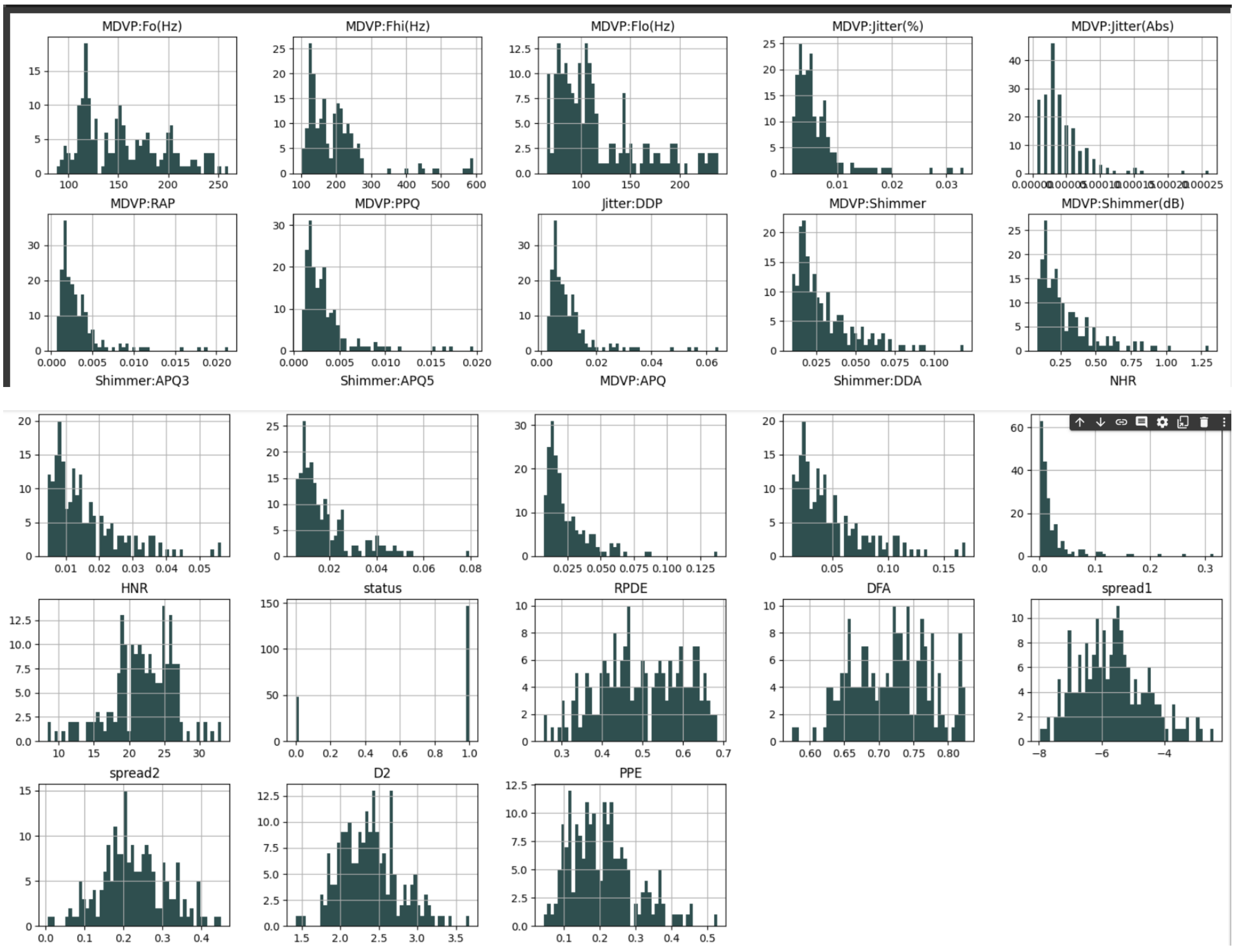
3.3) For low(Minimum) frequency



Observations:

From the above figures we can observe for both average and minimum vocal frequencies that the higher frequency have the higher tendency to belong to Parkinson Negative cases than of Positive cases. However, the same tendency is not valid for the maximum vocal frequency category.

Histogram



Observations:

- We can see some of the data is normally distributed and most of the attributes are right skewed

Boxplot

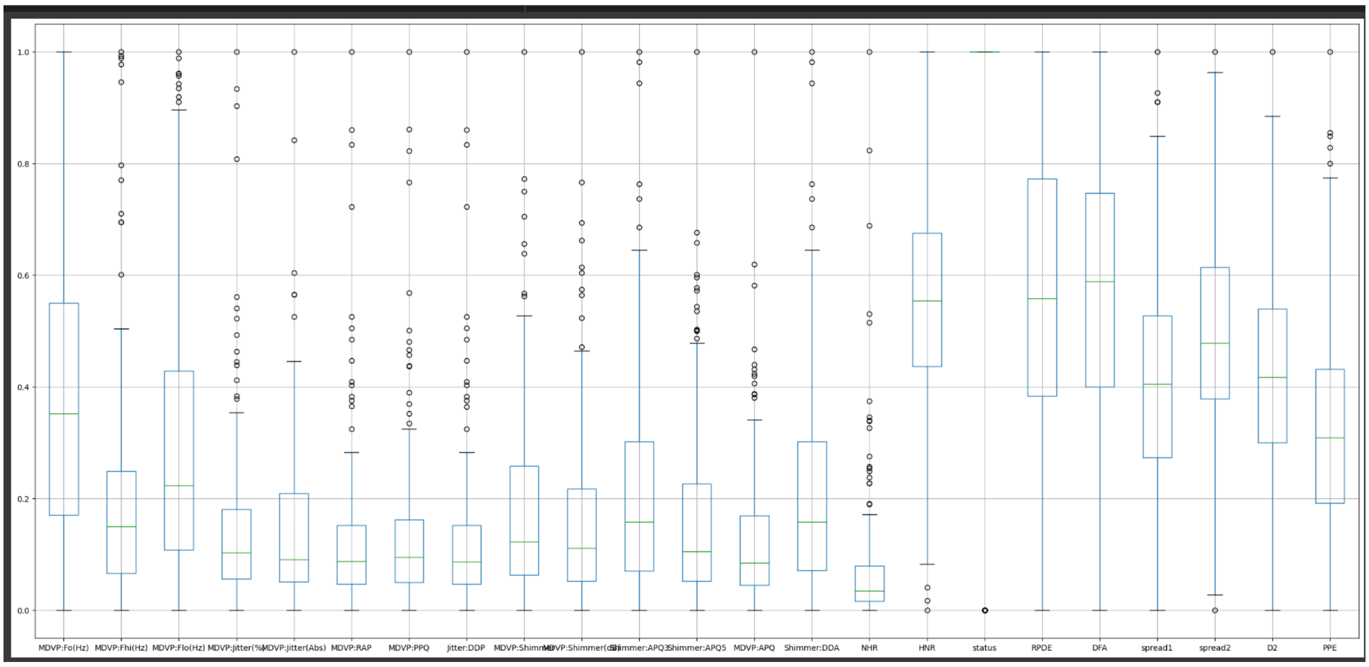
The input features were scaled before making boxplot

Observations from below Boxplot

- These points outside the whiskers are identified as outliers.

- Outliers are data points that fall outside the interquartile range (IQR) of the data and are often considered as extreme values. While outliers are commonly viewed as anomalies or errors, they can also represent real-world variability and provide valuable insights into the data.

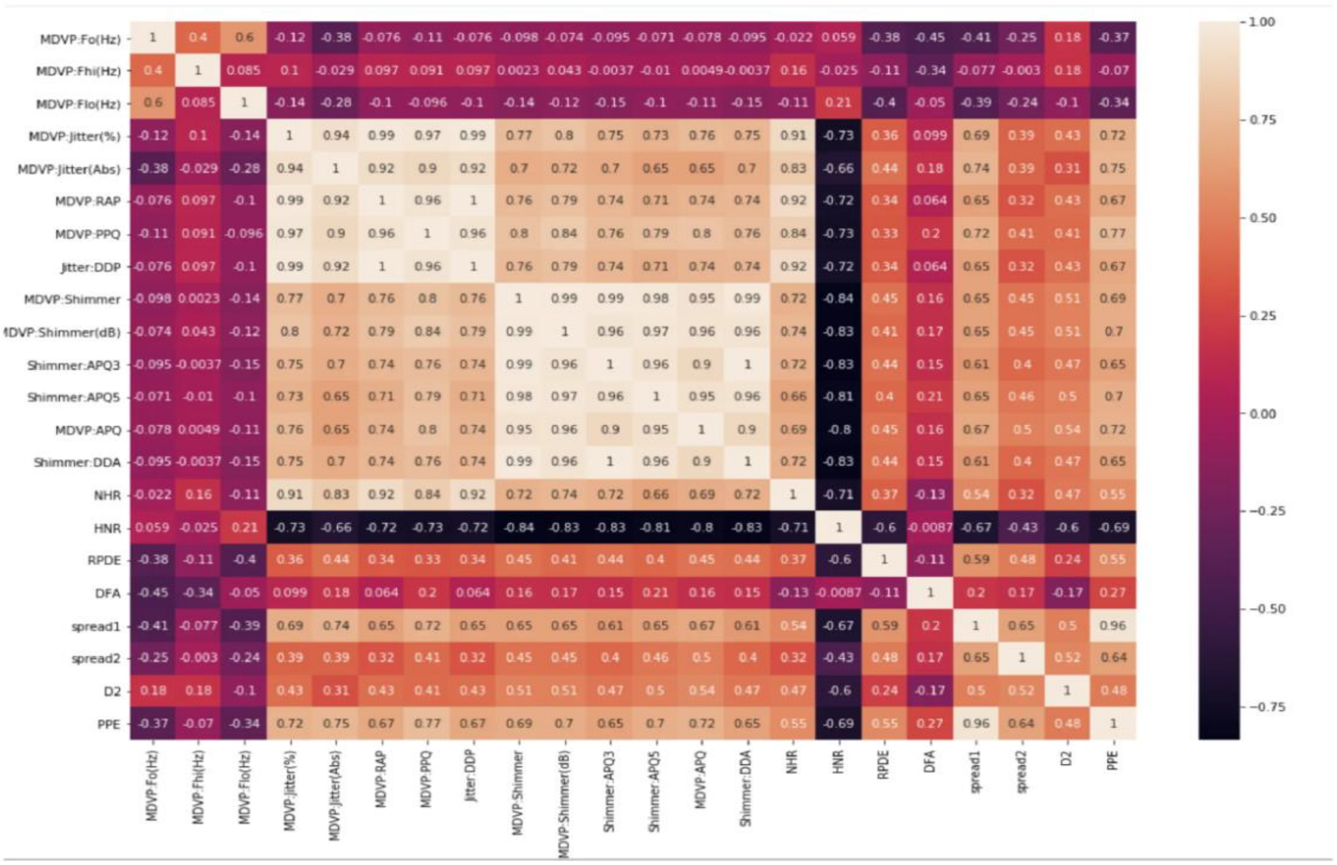
- According to our case and dataset distribution, outliers represent rare events or extreme cases that are important to include in the analysis in finance, outliers provide important insights into the data and the underlying distribution.

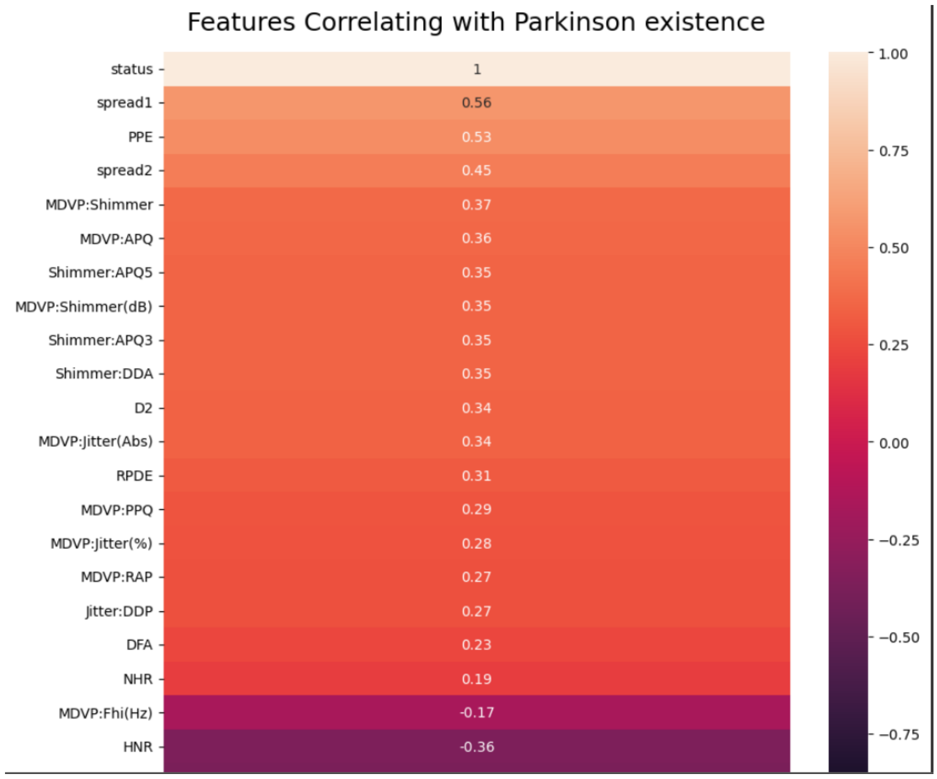


Bivariate Analysis

Correlation Plot

- As we can see in the below heatmap, features MDVP:Jitter(%) and MDVP:Shimmer have a strong positive correlation, and features MDVP:Jitter(%) and HNR have a strong negative correlation.
- Most of the features are highly correlated with each other implying that we can do dimension reduction like PCA in our analysis





This correlation plot shows features that are correlated with the response variable according to their decreasing order of positive correlation. It will help us to identify best features during feature selection .

Chapter – 4

Data Preprocessing

Splitting Data

Before developing the ML models, let’s standardize each feature in the dataset. We will use the StandardScaler class provided by the Sklearn library in Python. Further, we will split the input data into training and testing data with a 70:30 ratio.

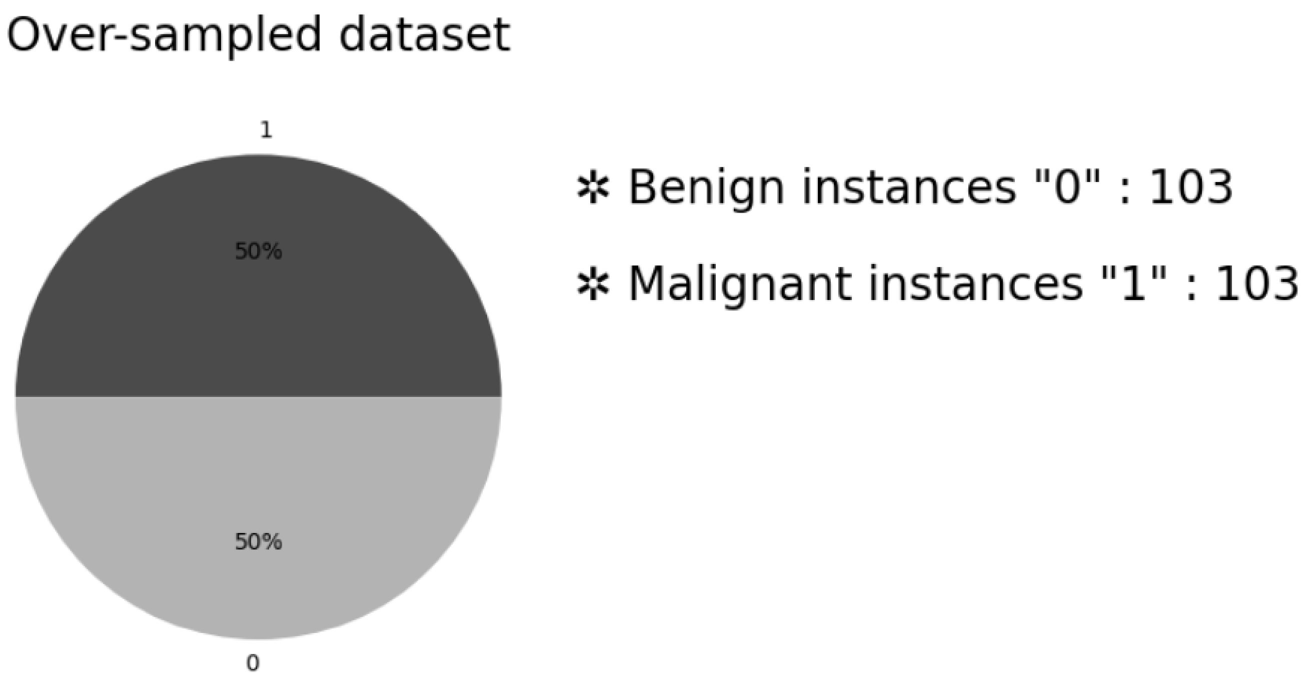
Feature Engineering

X will refer to our input features.

y will refer to the target feature[status].

Steps:

- We will now split the data into 70:30 that is train and test data
- After that we will standardize train data and test data
- Since we have observed that there was mild class imbalance in our dataset
  We will perform our analysis with balancing dataset and without balance and compare our results



Chapter 5

Modelling

1st Scenario : where the models will be trained and evaluated on the unbalanced dataset.

As observed earlier in our dataset, certain variables exhibit a wide range of values, with some being exceptionally large and others exceedingly small. To address this, it is advisable to standardize our features before proceeding with model development. Consequently, we have standardized the training data. Following this standardization process, the training data now appears as follows:

K-NN

In k-NN, the basic idea is to predict the label of a new instance based on the labels of its k nearest neighbours in the training data.

The goal of the k-nearest neighbour algorithm is to identify the nearest neighbours of a given query point, so that we can assign a class label to that point. In order to do this, KNN has a few requirements:

Step-1: Select the number K of the neighbours

Step-2: Calculate the Euclidean distance of K number of neighbours

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbours, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbour is maximum.

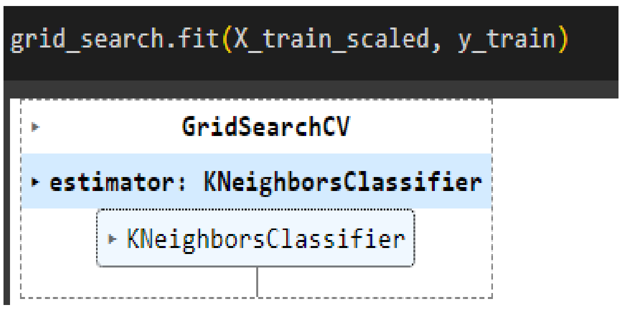
Step-6: Our model is ready.

Grid Search

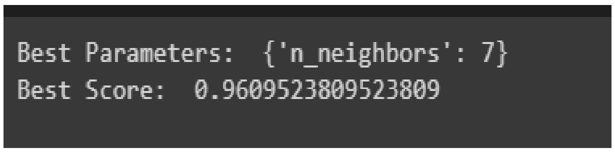
In medical testing, recall is a measure of the ability of a test to correctly identify true positive cases, or the proportion of actual positive cases that are correctly identified by the test. A high recall means that the test is able to correctly identify a large proportion of true positive cases, even if it may also produce some false positive results, therefore for scoring we select recall . The param_grid parameter specifies a dictionary of hyperparameters to search over, while the scoring parameter specifies the performance metric to optimize for, which in this case is the recall.

Hyperparamter Tuning:

Since we know that the parameter "k" in the KNN is a hyperparameter, so we must tune it, therefore, for tuning we have used the 5-fold Kross validation.

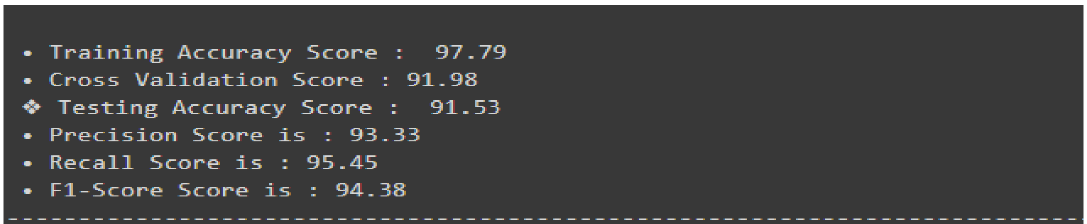


```
grid_search.fit(X_train_scaled, y_train)
```

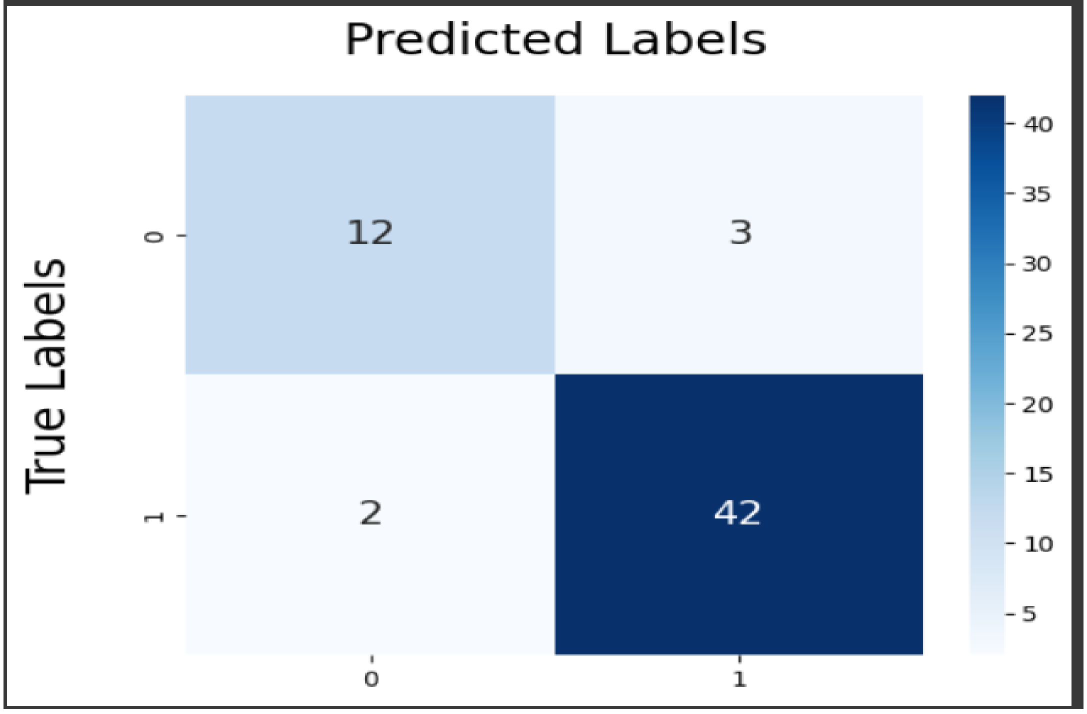


```
Best Parameters:  {'n_neighbors': 7}
Best Score:  0.9609523809523809
```

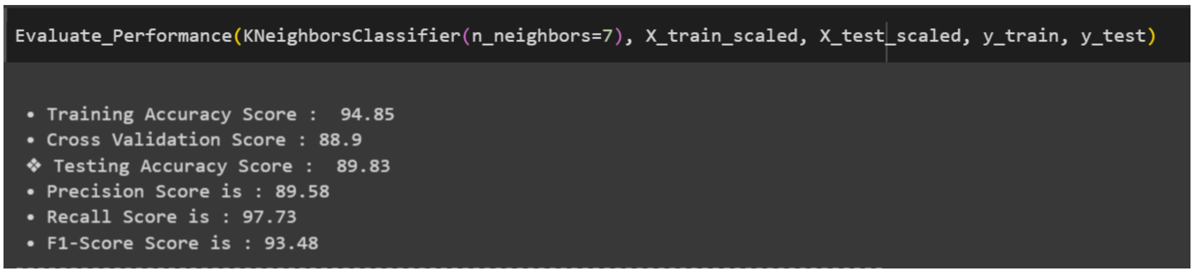
Now since we have obtained best parameter for K-nearest neighbour we will train the model using this K value and calculated the classification report on the test data



```
• Training Accuracy Score :   97.79
• Cross Validation Score : 91.98
❖ Testing Accuracy Score :   91.53
• Precision Score is : 93.33
• Recall Score is : 95.45
• F1-Score Score is : 94.38
```



Report using best parameter using Grid search



```
Evaluate_Performance(KNeighborsClassifier(n_neighbors=7), X_train_scaled, X_test_scaled, y_train, y_test)

• Training Accuracy Score :   94.85
• Cross Validation Score : 88.9
❖ Testing Accuracy Score :   89.83
• Precision Score is : 89.58
• Recall Score is : 97.73
• F1-Score Score is : 93.48
```

Logistic Regression

This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.

- Prepare the data: The data should be in a format where each row represents a single observation and each column represents a different variable. The target variable (the variable you want to predict) should be binary (yes/no, true/false, 0/1).
- Train the model: We teach the model by showing it the training data. This involves finding the values of the model parameters that minimize the error in the training data.
- Evaluate the model: The model is evaluated on the held-out test data to assess its performance on unseen data.
- Use the model to make predictions: After the model has been trained and assessed, it can be used to forecast outcomes on new data.

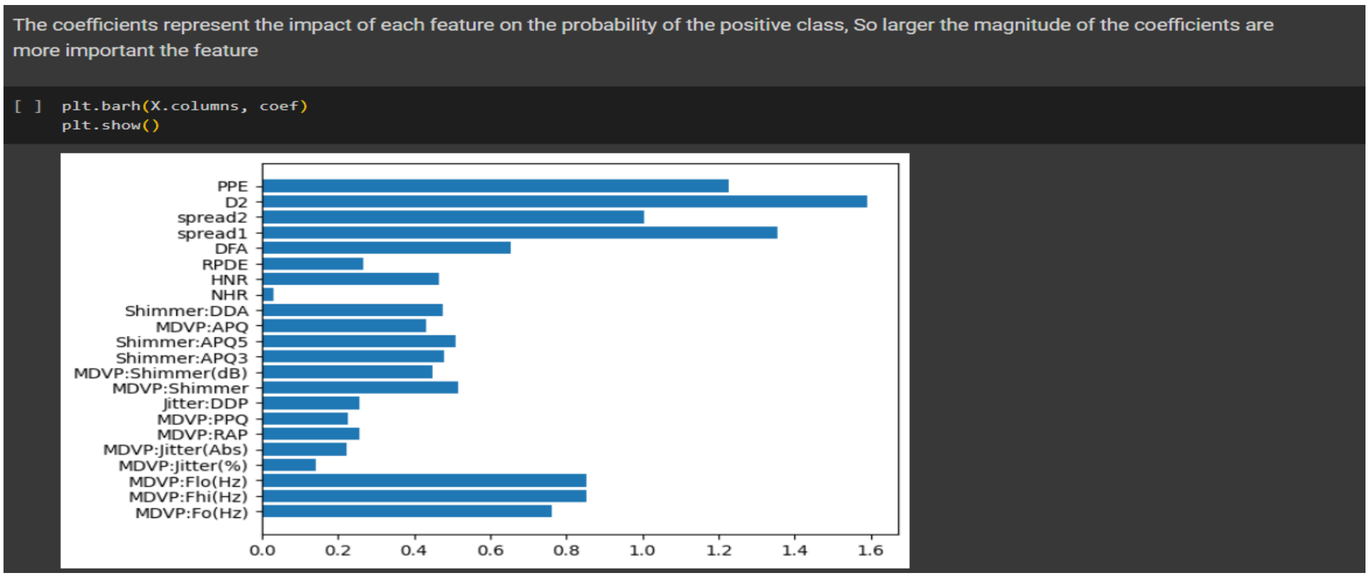
$$\text{Log loss} = \frac{1}{N} \sum_{i=1}^{N} -(y_i * \log(\hat{Y}_i) + (1-y_i) * \log(1-\hat{Y}_i))$$

Grid Search

While grid search can be used to optimize hyperparameters for many machine learning algorithms, the hyperparameters in logistic regression may not be particularly insightful. Instead, we can use logistic regression to identify feature importance, as the coefficients learned during training can reveal which features are most strongly associated with the target variable. We can obtain the coefficients.

AUC-ROC Curve

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.

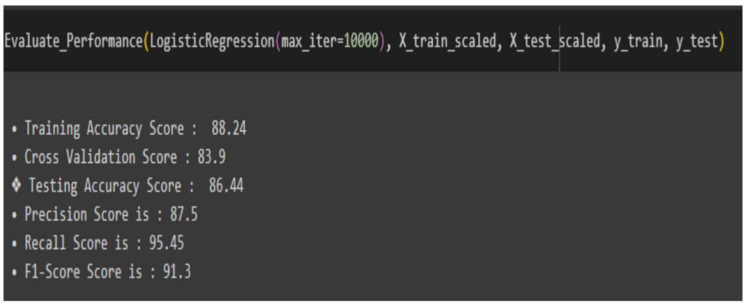

The coefficients represent the impact of each feature on the probability of the positive class, So larger the magnitude of the coefficients are more important the feature

```
[ ]  plt.barh(X.columns, coef)
     plt.show()
```

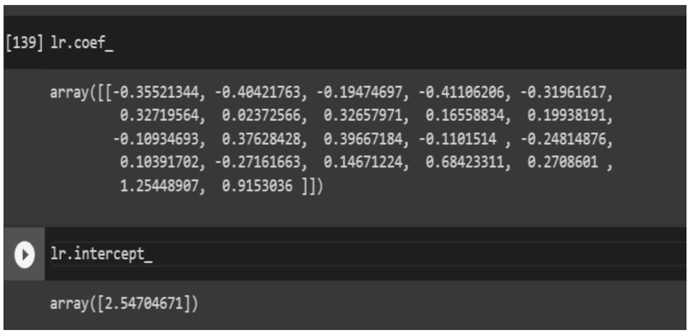
We can select the features with the highest magnitude coefficients

We see many of the features are redundant, identifying and removing redundant features can potentially help in a few ways. First, it can reduce the amount of data required for the model, which can reduce the cost of storing and processing data. Second, it can improve the performance of the model, allowing it to make more accurate predictions with fewer resources. This can help reduce the cost of misclassifications or errors in the predictions. Finally, it can also reduce the cost of model development and maintenance. Removing redundant features can simplify the model, making it easier to develop and maintain over time. It can also reduce the need for frequent updates and retraining, which can be costly in terms of time and resources

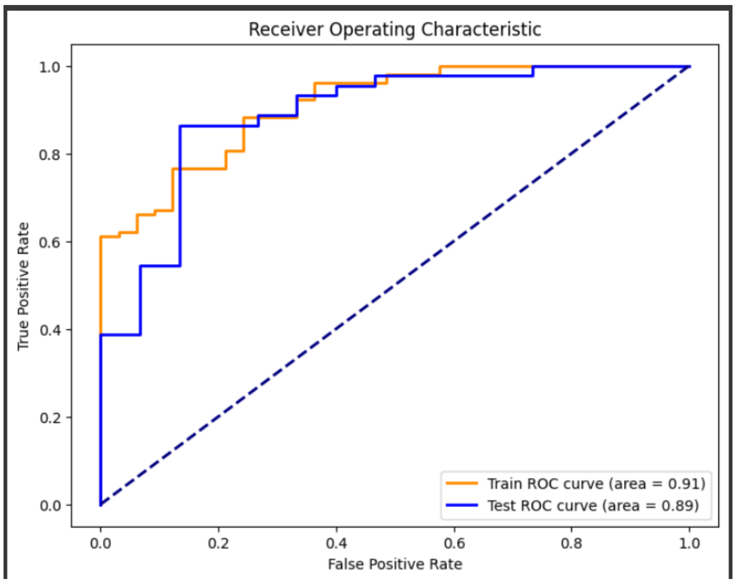
Classification Report for logistic regression



```
Evaluate_Performance(LogisticRegression(max_iter=10000), X_train_scaled, X_test_scaled, y_train, y_test)

• Training Accuracy Score :  88.24
• Cross Validation Score : 83.9
• Testing Accuracy Score :  86.44
• Precision Score is : 87.5
• Recall Score is : 95.45
• F1-Score Score is : 91.3
```



```
[139] lr.coef_

      array([[-0.35521344, -0.40421763, -0.19474697, -0.41106206, -0.31961617,
               0.32719564,  0.02372566,  0.32657971,  0.16558834,  0.19938191,
              -0.10934693,  0.37628428,  0.39667184, -0.1101514 , -0.24814876,
               0.10391702, -0.27161663,  0.14671224,  0.68423311,  0.2708601 ,
               1.25448907,  0.9153036 ]])

      lr.intercept_

      array([2.54704671])
```



AUC-ROC Curve

Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease

Decision Tree Classifier:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

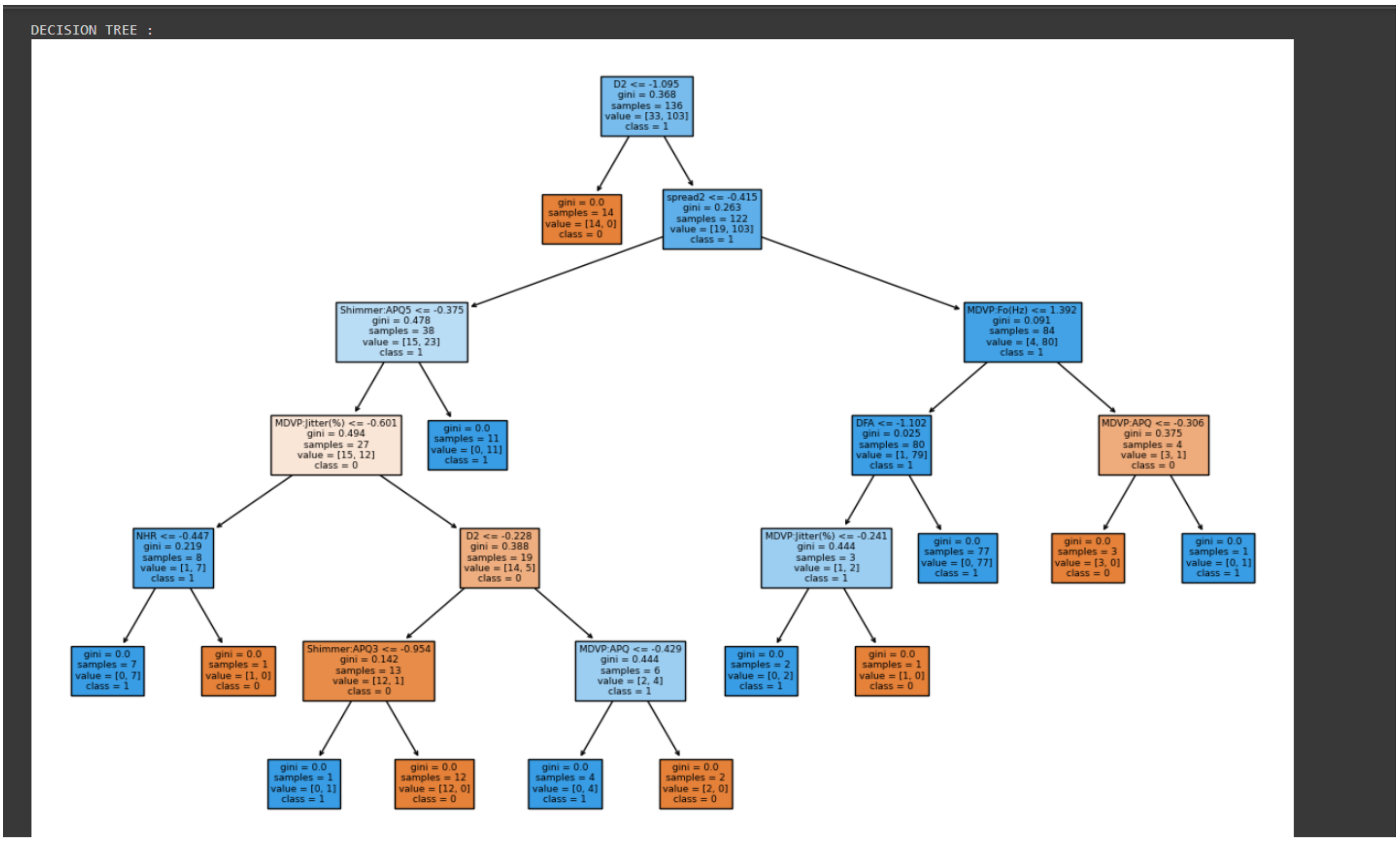
Step-2: Find the best attribute in the dataset

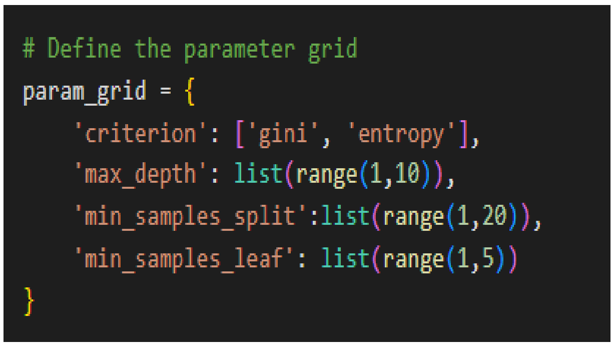
Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

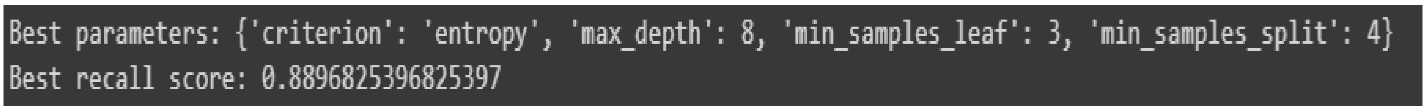
A fully grown decision tree is shown below in the plot


DECISION TREE :
D2 <= -1.095 gini = 0.368 samples = 136 value = [33, 103] class = 1
gini = 0.0 samples = 14 value = [14, 0] class = 0
spread2 <= -0.415 gini = 0.263 samples = 122 value = [19, 103] class = 1
Shimmer:APQ5 <= -0.375 gini = 0.478 samples = 38 value = [15, 23] class = 1
MDVP:Fo(Hz) <= 1.392 gini = 0.091 samples = 84 value = [4, 80] class = 1
MDVP:Jitter(%) <= -0.601 gini = 0.494 samples = 27 value = [15, 12] class = 0
gini = 0.0 samples = 11 value = [0, 11] class = 1
DFA <= -1.102 gini = 0.025 samples = 80 value = [1, 79] class = 1
MDVP:APQ <= -0.306 gini = 0.375 samples = 4 value = [3, 1] class = 0
NHR <= -0.447 gini = 0.219 samples = 8 value = [1, 7] class = 1
D2 <= -0.228 gini = 0.388 samples = 19 value = [14, 5] class = 0
MDVP:Jitter(%) <= -0.241 gini = 0.444 samples = 3 value = [1, 2] class = 1
gini = 0.0 samples = 77 value = [0, 77] class = 1
gini = 0.0 samples = 3 value = [3, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 7 value = [0, 7] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
Shimmer:APQ3 <= -0.954 gini = 0.142 samples = 13 value = [12, 1] class = 0
MDVP:APQ <= -0.429 gini = 0.444 samples = 6 value = [2, 4] class = 1
gini = 0.0 samples = 2 value = [0, 2] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 12 value = [12, 0] class = 0
gini = 0.0 samples = 4 value = [0, 4] class = 1
gini = 0.0 samples = 2 value = [2, 0] class = 0

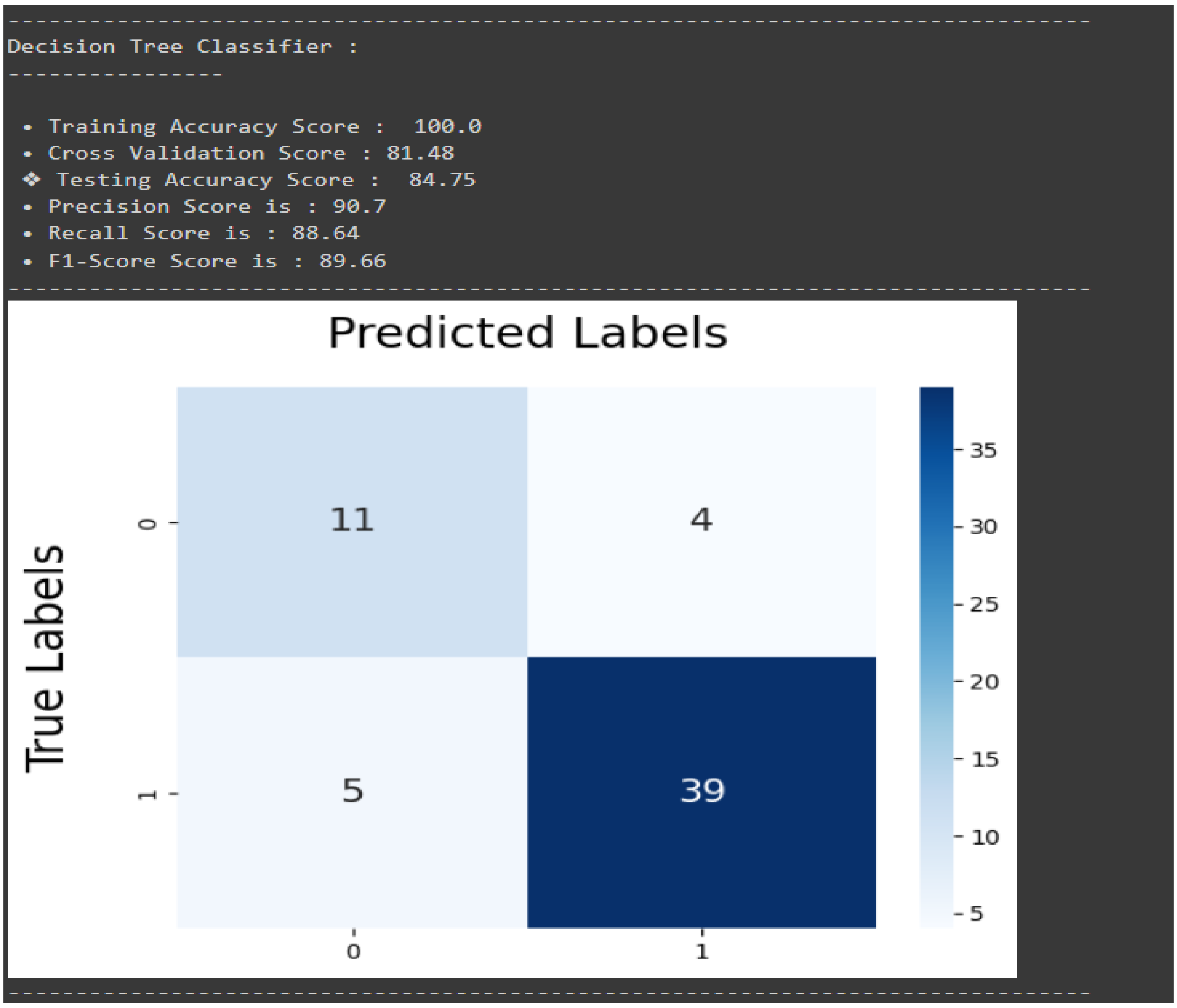


```
# Define the parameter grid
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': list(range(1,10)),
    'min_samples_split':list(range(1,20)),
    'min_samples_leaf': list(range(1,5))
}
```

Classification report for fully grown decision tree is given below. Applying Grid search over hyperparameter to do post pruning to prevent overfitting



```
Best parameters: {'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 3, 'min_samples_split': 4}
Best recall score: 0.8896825396825397
```



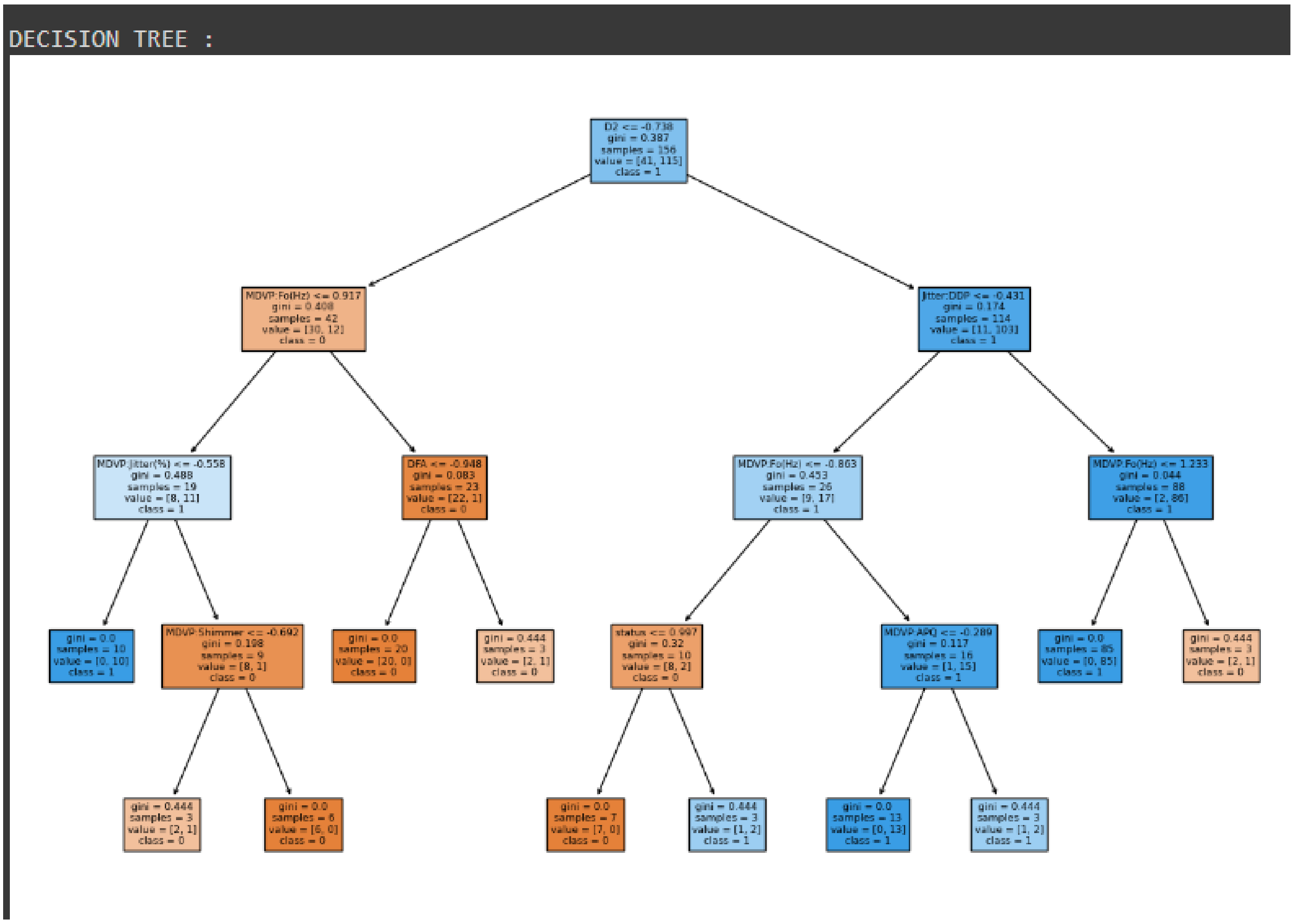
```
--------------------------------------------------------------------------------
Decision Tree Classifier :
----------------

 • Training Accuracy Score :  100.0
 • Cross Validation Score : 81.48
 ❖ Testing Accuracy Score :  84.75
 • Precision Score is : 90.7
 • Recall Score is : 88.64
 • F1-Score Score is : 89.66
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
```

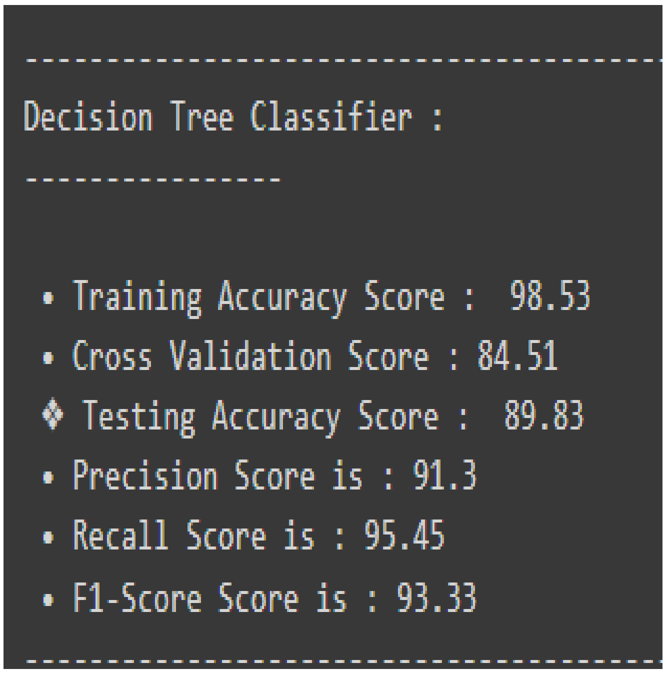
Post Pruned Decision Tree

Post-pruning (or just pruning) is the most common way of simplifying trees. Here, nodes and subtrees are replaced with leaves to reduce complexity. Pruning can not only significantly reduce the size but also improve the classification accuracy of unseen objects. It may be the case that the accuracy of the assignment on the train set deteriorates, but the accuracy of the classification properties of the tree increases overall.

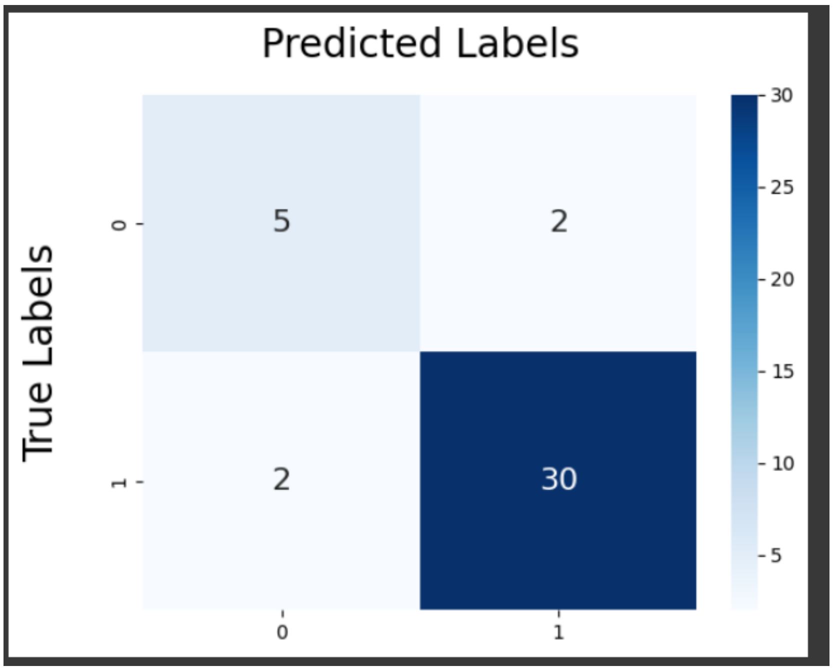
The procedures are differentiated on the basis of their approach in the tree (top-down or bottom-up).

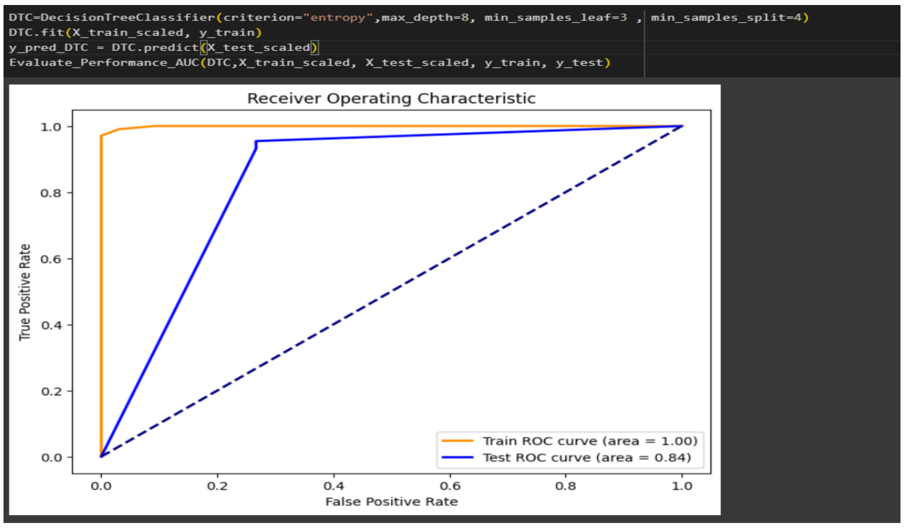


DECISION TREE :



```
----------------------------------------
Decision Tree Classifier :
----------------

 • Training Accuracy Score :  98.53
 • Cross Validation Score : 84.51
 ◈ Testing Accuracy Score :  89.83
 • Precision Score is : 91.3
 • Recall Score is : 95.45
 • F1-Score Score is : 93.33
----------------------------------------
```

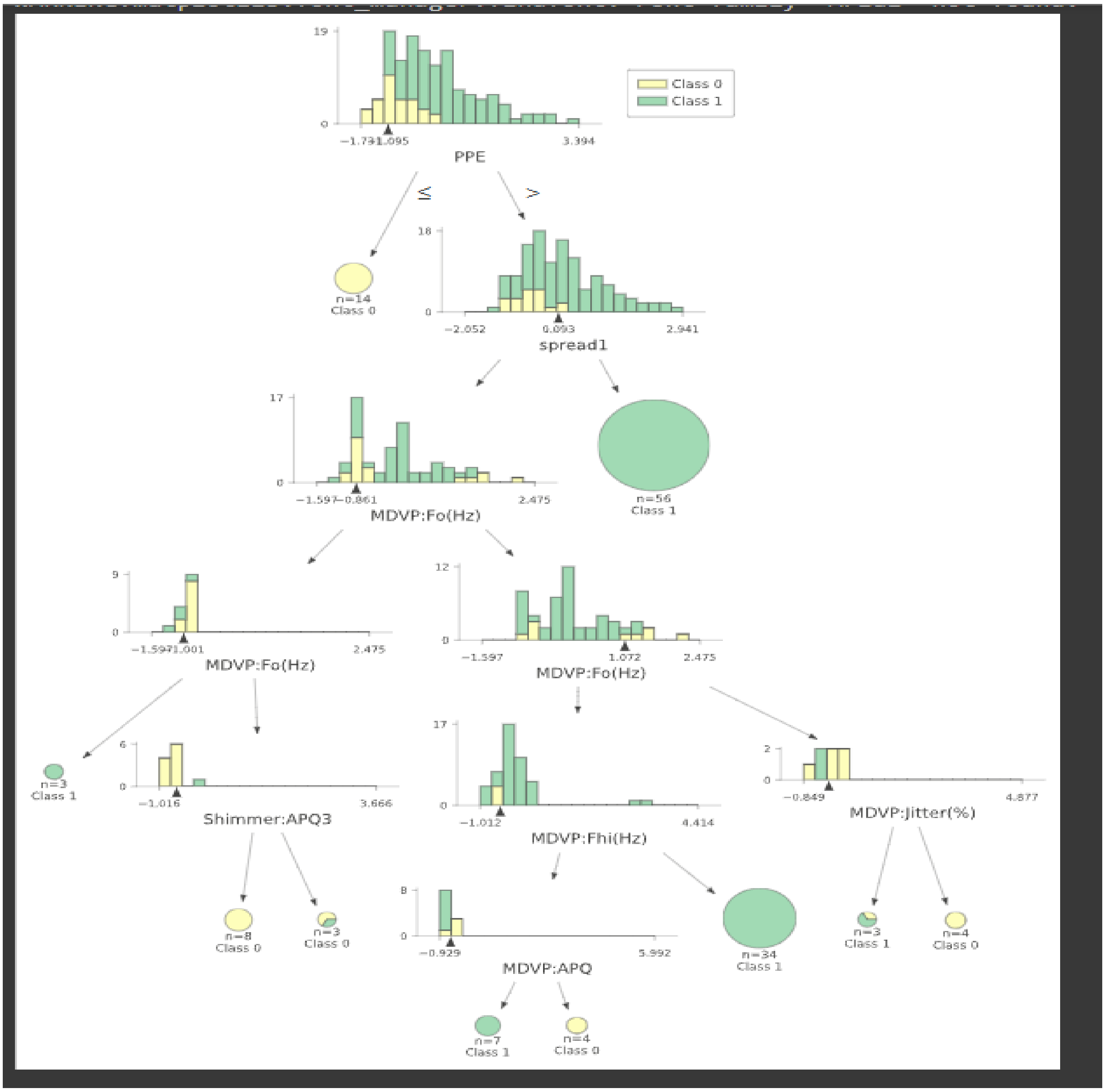




```
DTC=DecisionTreeClassifier(criterion="entropy",max_depth=8, min_samples_leaf=3 , min_samples_split=4)
DTC.fit(X_train_scaled, y_train)
y_pred_DTC = DTC.predict(X_test_scaled)
Evaluate_Performance_AUC(DTC,X_train_scaled, X_test_scaled, y_train, y_test)
```

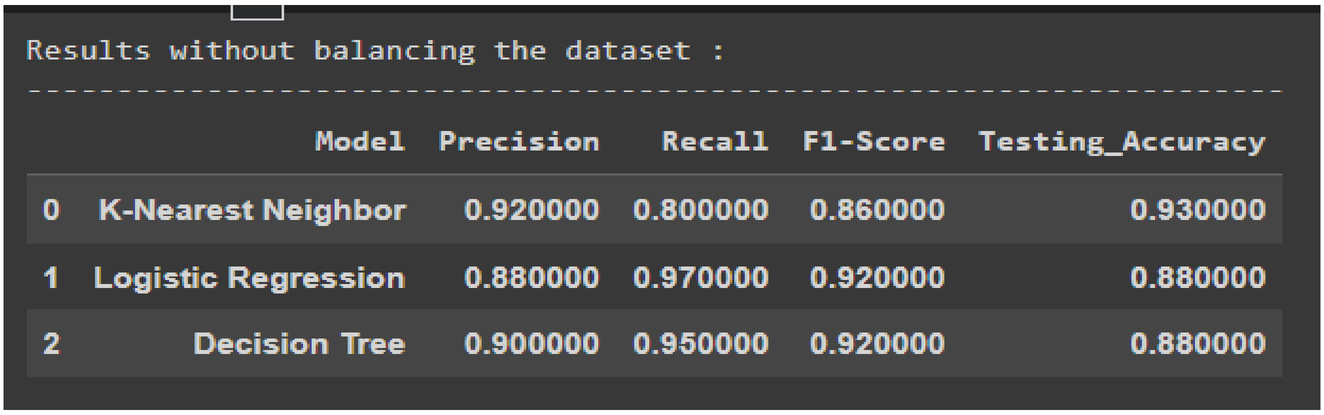
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Train ROC curve (area = 1.00)
Test ROC curve (area = 0.84)

Decision Tree Plot is shown below



Comparison Table

Table for comparison between Decision Tree , K-NN , Logistic Regression without balancing dataset



```
Results without balancing the dataset :
----------------------------------------------------------------------
```

	Model	Precision	Recall	F1-Score	Testing_Accuracy
0	K-Nearest Neighbor	0.920000	0.800000	0.860000	0.930000
1	Logistic Regression	0.880000	0.970000	0.920000	0.880000
2	Decision Tree	0.900000	0.950000	0.920000	0.880000

✳ 2nd Scenario : where the models will be trained and evaluated on the oversampled dataset

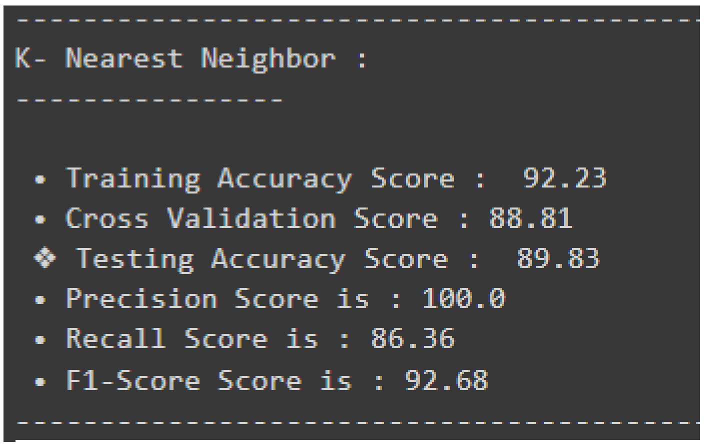
As shown in the above figure, the dataset appears to be unbalanced, with a significantly higher number of samples in Malignant category compared to the other.

To address the issue of class imbalance, a over-sampling data balancing technique will be used

Oversampling technique

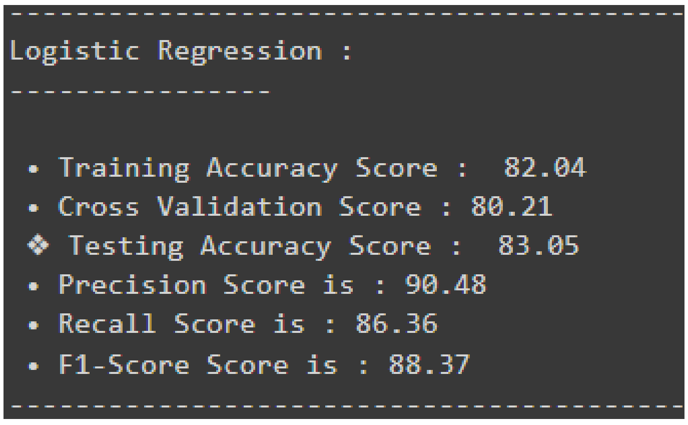
The Over Sampler technique is a oversampling used to address class imbalance, Oversampling involves increasing the number of samples in the minority class to match the number of samples in the majority class.

K-Nearest Neighbour



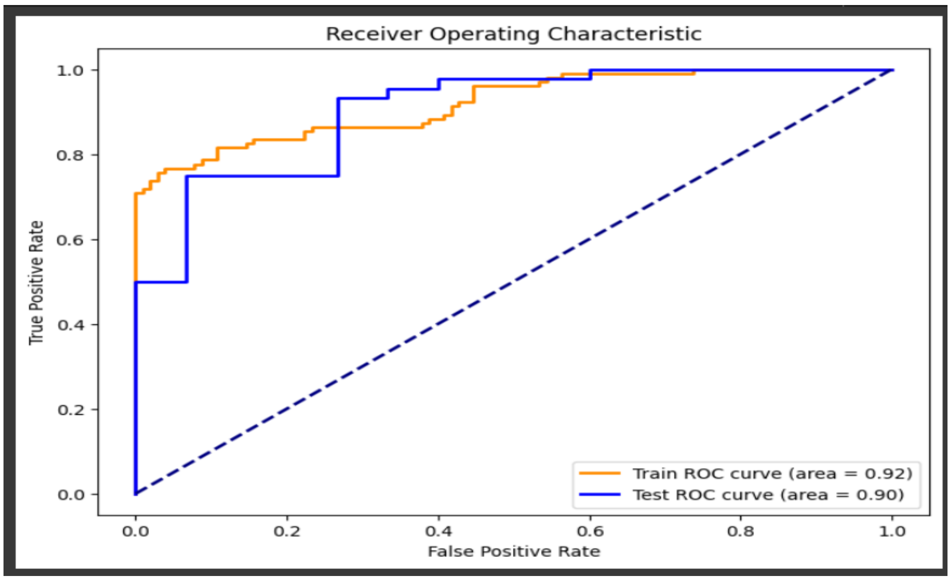
```
----------------------------------------
K- Nearest Neighbor :
----------------

 • Training Accuracy Score :  92.23
 • Cross Validation Score : 88.81
 ❖ Testing Accuracy Score :  89.83
 • Precision Score is : 100.0
 • Recall Score is : 86.36
 • F1-Score Score is : 92.68
----------------------------------------
```

Logistic Regression



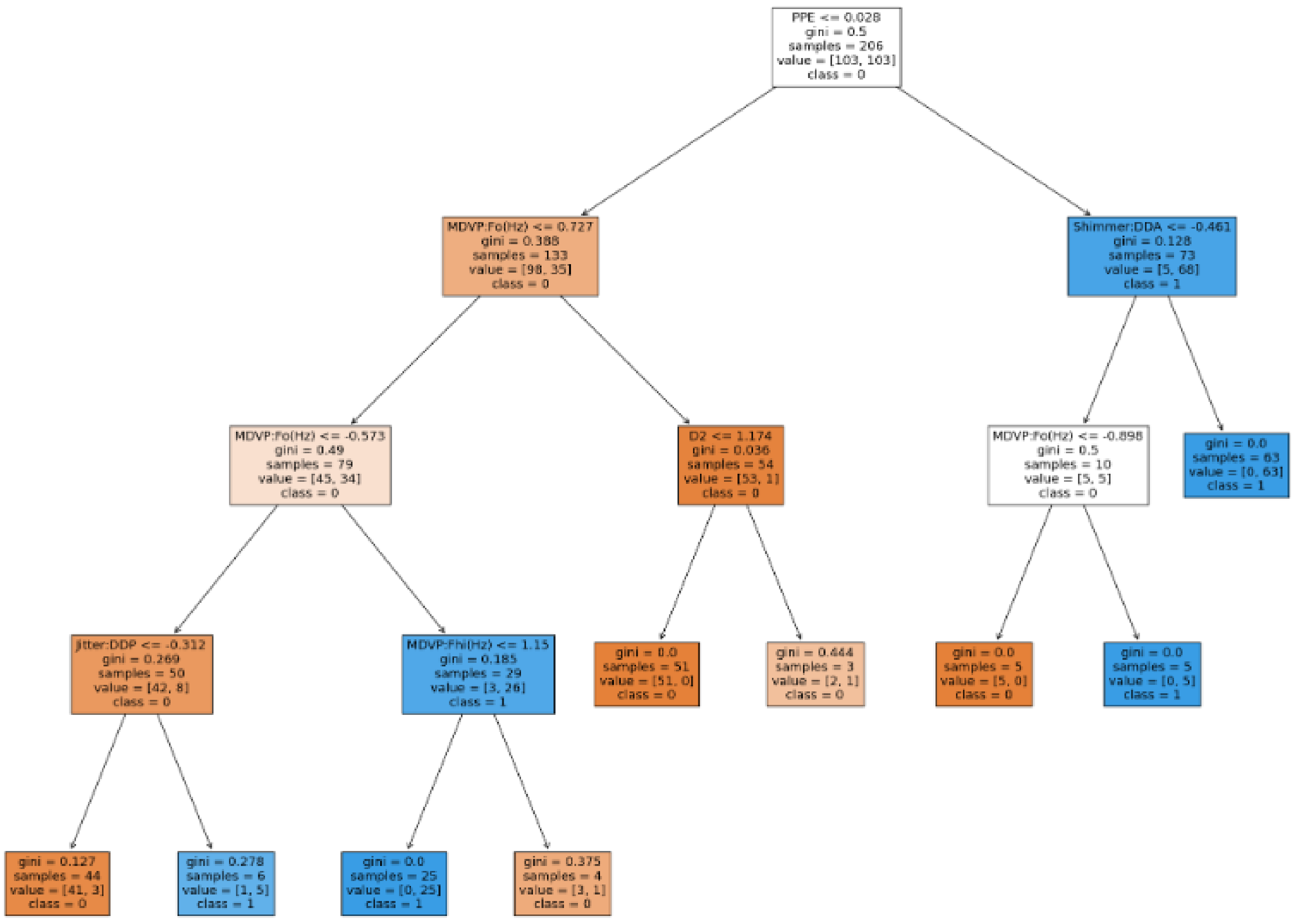
```
----------------------------------------
Logistic Regression :
----------------

 • Training Accuracy Score :  82.04
 • Cross Validation Score : 80.21
 ❖ Testing Accuracy Score :  83.05
 • Precision Score is : 90.48
 • Recall Score is : 86.36
 • F1-Score Score is : 88.37
----------------------------------------
```

AUC ROC Curve Logistic Regression in case of Balanced Data

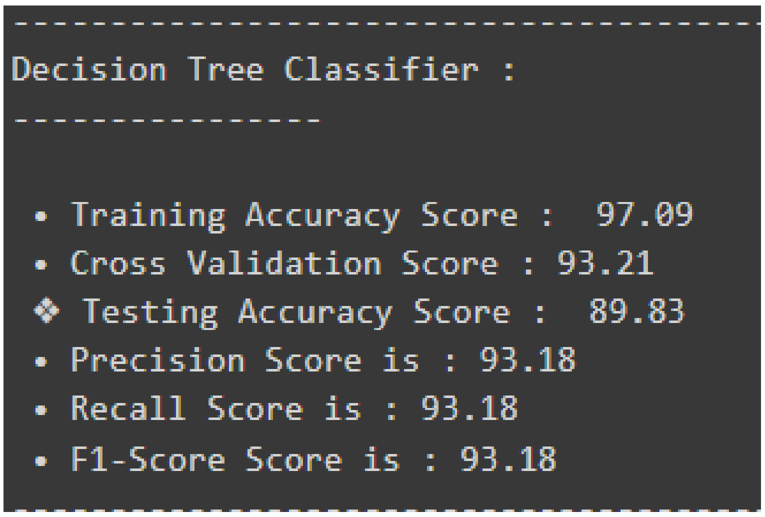


Decision Tree

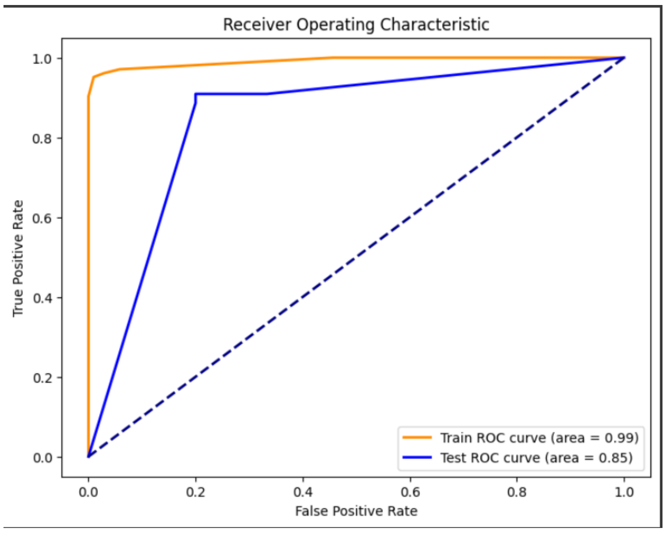
Result obtained after performing oversampling to the minority class and fitting decision tree



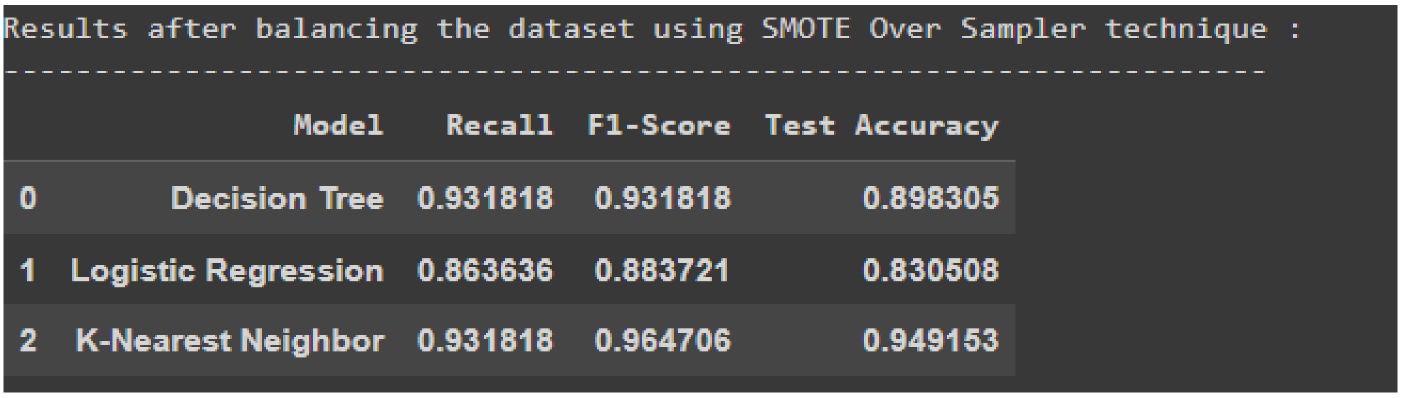
Classification report for decision tree based on oversampled data



```
----------------------------------------
Decision Tree Classifier :
----------------

 • Training Accuracy Score :  97.09
 • Cross Validation Score : 93.21
 ❖ Testing Accuracy Score :  89.83
 • Precision Score is : 93.18
 • Recall Score is : 93.18
 • F1-Score Score is : 93.18
----------------------------------------
```



Comparison Table for Model based on Balanced Data



Results after balancing the dataset using SMOTE Over Sampler technique :

	Model	Recall	F1-Score	Test Accuracy
0	Decision Tree	0.931818	0.931818	0.898305
1	Logistic Regression	0.863636	0.883721	0.830508
2	K-Nearest Neighbor	0.931818	0.964706	0.949153

Chapter 5

Principal Component Analysis

Let's see by visualization impact of doing PCA with and without standardization then we will move on to find principal components

A Principal Component Analysis is concerned with explaining the Variance-Co-Variance structure through a few linear combinations of the original variables.

Its general objective is:

1) Dimensionality Reduction 2) Interpretation

$$Let\ a\ random\ vector\ X' = [X_1, X_2, \ldots., X_p\ ]\ \ have\ the\ Covariance\ Matrix \sum$$

with eigenvalues λ_1≥λ_2≥⋯.≥λ_p≥0,then we have the Principal Components as:

$$Y_1 = l_1'\,X = l_{11}\,X_1 + l_{21}\,X_2 + \cdots + l_{p1}\,X_p$$

$$Y_2 = l_2'\,X = l_{12}\,X_1 + l_{22}\,X_2 + \cdots + l_{p2}\,X_p$$

....

$$Y_p = l_p'\,X = l_{1p}\,X_1 + l_{2p}\,X_2 + \cdots + l_{pp}\,X_p$$

Are the p Principal Components.

$Where\ l\_1, l\_2, \ldots, l\_p\ \ are\ the\ p\ Normalize\ Eigen\ Vectors\ Corresponding\ to\ the\ p\ eigen\ Values\ \lambda\_1, \ldots, \lambda\_p$

Also Note that:

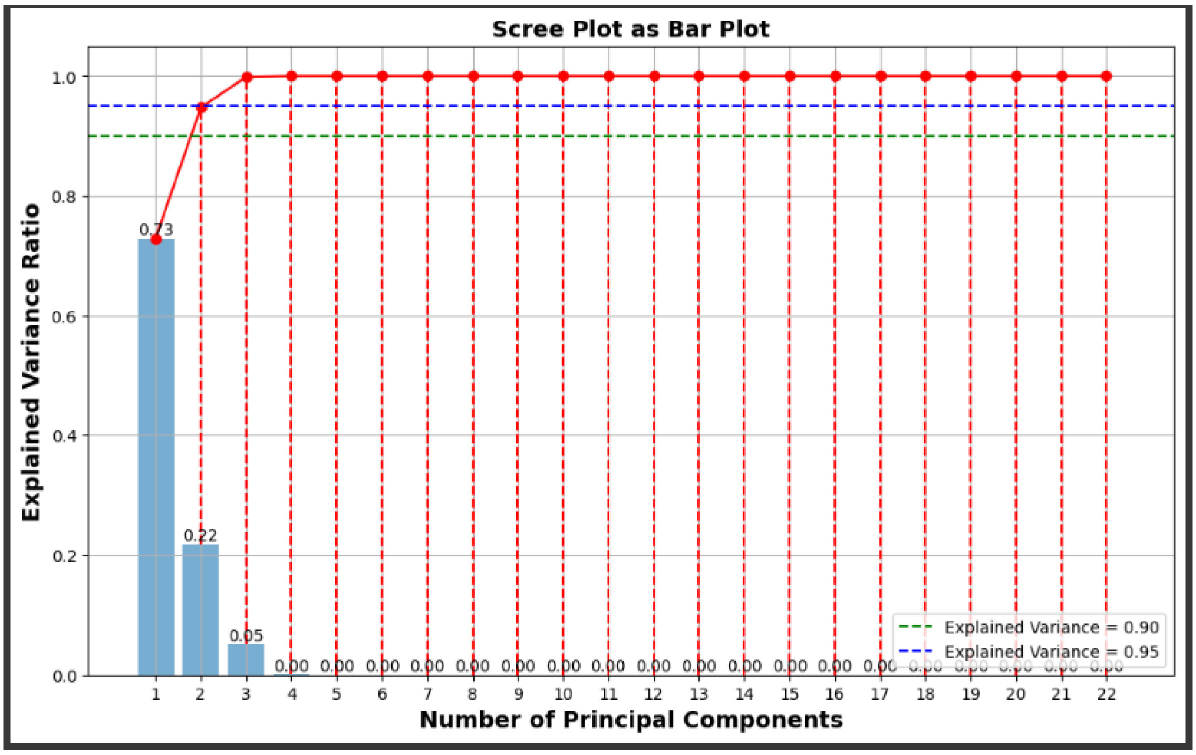
$$Var(Y_i) = l_i'\sum l_i\ \ = \lambda_i \qquad for\ i = 1,2,3, \ldots, p\ \ and$$

$$Cov(Y_i, Y_k) = l_i'\sum l_k\ \ = 0 \quad for\ i\ \neq\ k = 1,2,3, \ldots, p$$

Proportion of total population variance due to the k^th principal component=

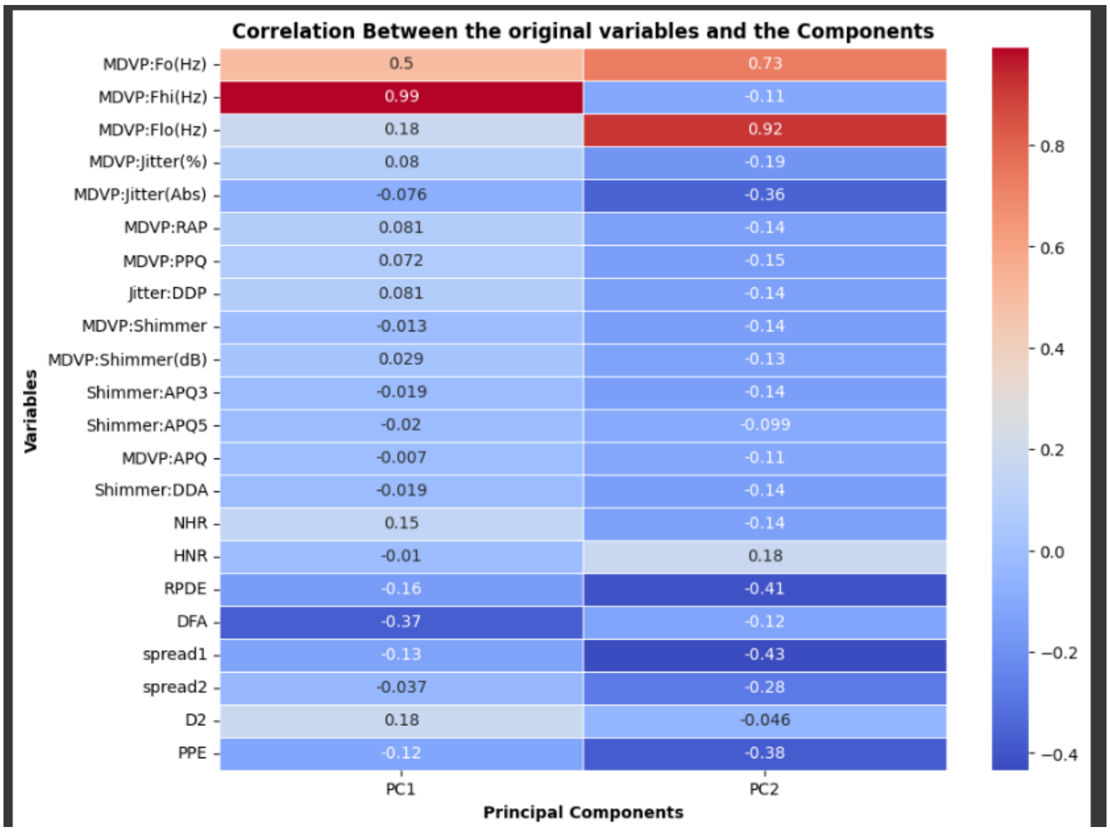
$$\lambda_k/(\lambda_1 + \lambda_2 + \cdots + \lambda_p\ )$$

Before Scaling:



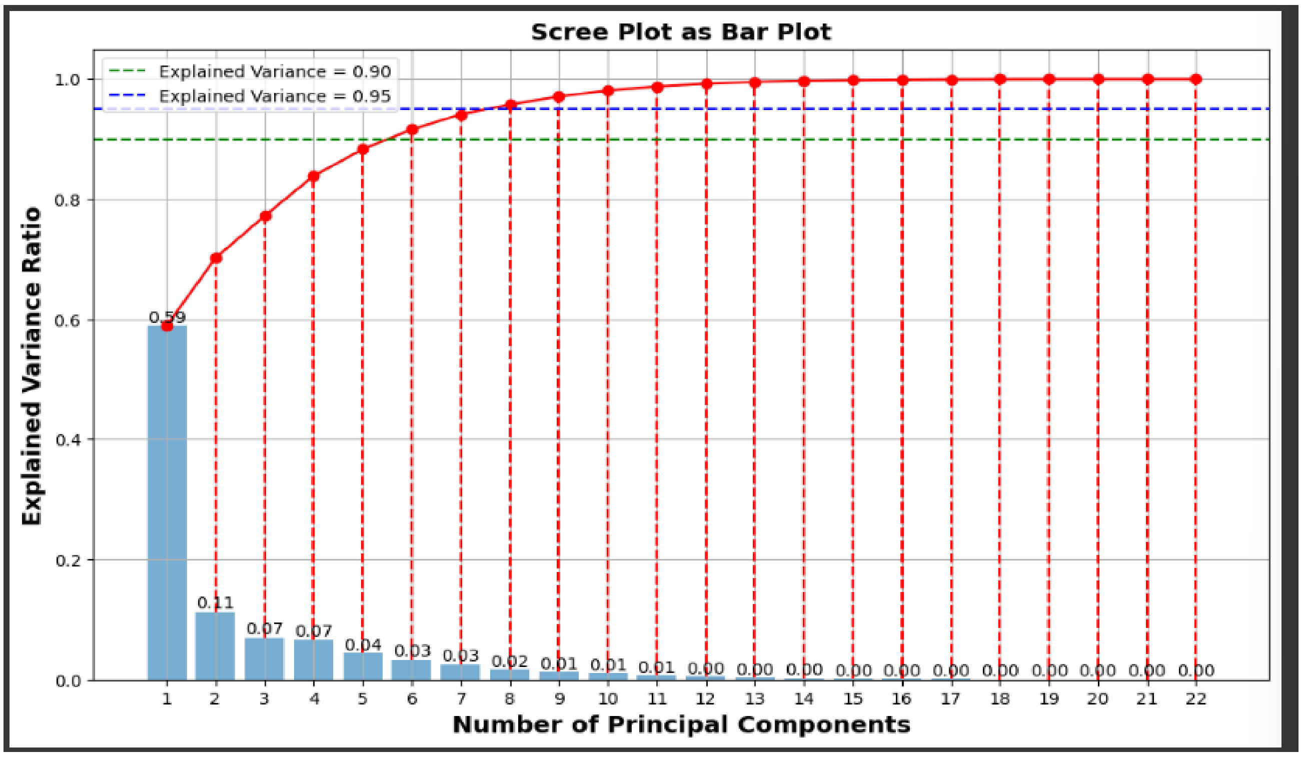
From the above plot we can see that approximately 90% of the variance was explained by only first two principal components this was due to fact of high variability in the range of values taken by different columns of this dataset that is some values are 100 range while some where is 10^-3 range so that creates high variability in columns apart from other columns that’s why we can see that majority of the variance was explained by first two components only .

Below we have also calculated the correlation between the original variable and the components to know which variable has high weightage on which components



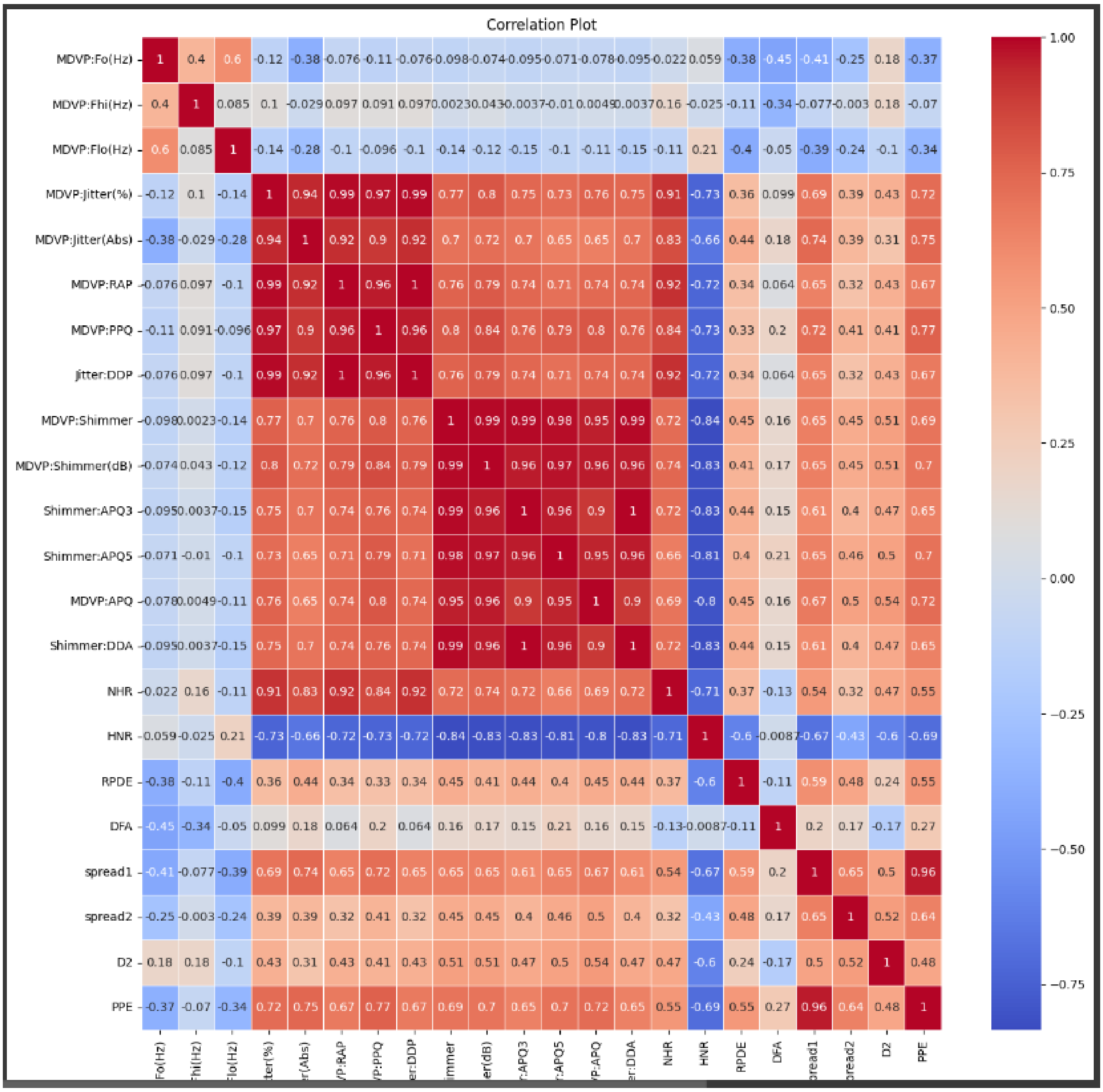
After Scaling

Now we will do scaling to all the columns so that each column comes in some range then we will apply pca and see the results from the scree plot



From the above plot we can see that now no of components has increased from 2 to 5 as of now all values were in same range

The correlation plot is shown for the columns of the dataset



As we can see there is high correlation between various features of the dataset that is they aren't independent We want to apply naïve bayes algorithm to above dataset form classification but the assumption for naïve was features should be independent with respect to each class so we will apply pca on scaled variable then apply naïve bayes algorithm to classify the observation . As done above we will apply naïve bayes on both balanced and unbalanced dataset and compare the results

Naïve Bayes

Step 1: Now to perform The Naïve Bayes Classifier, we assume that within the $\boldsymbol{k\ th}$ class, the p predictors are independently distributed, i.e., we assume that:

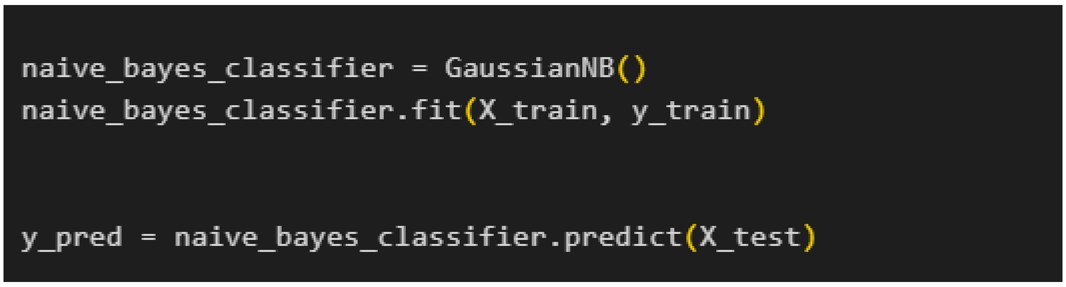
$\boldsymbol{fk}\ (x) = f\text{k}\ (x1\ ) * \text{fk}\ (x2\ ) * \ldots * fk\ (x16) * \text{fk}(\text{x}17);\ k = 0{,}1$

NOTE: For our data set we have assumed that: $fk(xj)$ ~N(μ_k, σ_k 2) ; k = 0,1 and j = 1,2,3, … , n_k

Step 2: Once, we made this assumption, we get the following posterior probability as:

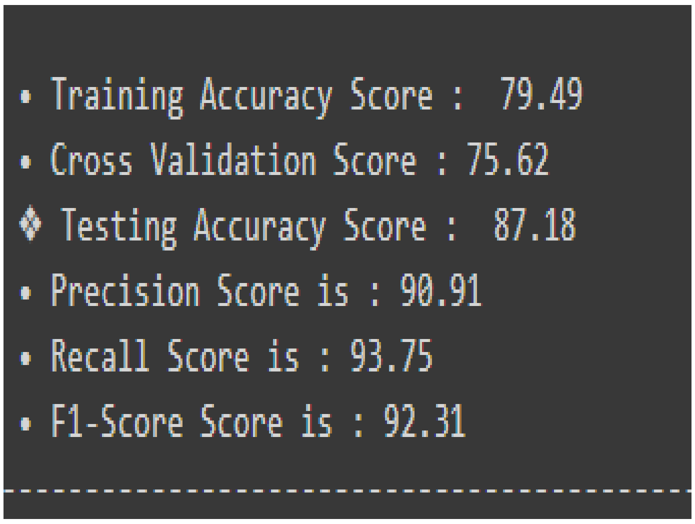
$\boldsymbol{P}(\boldsymbol{Y} = \boldsymbol{k}|\text{X} = \text{x}) = \frac{pk\, *\, fk\ (x1\ )*\ fk\ (x2\ )*\ \ldots\ *\ fkx16}{\sum i=0\ to\ 1(\text{pl} * \text{fl}\ (\text{x}1\ )* \text{fl}\ (\text{x}2\ )* \ldots * \text{fl}\ (\text{x}16))}$ l=0 ; $x = (x1, \ldots, x16, \text{x}17)^{\text{T}}\ and$ k = 0,1

Step 3: The Decision Rule: A given observation $\boldsymbol{x} = (\boldsymbol{x1}, \boldsymbol{x}2, \ldots, \text{x}16,\text{x}17)^{T}$ belongs/classified in to the k th class for which the posterior probability $P(Y = k|X = x)$ is Maximum.

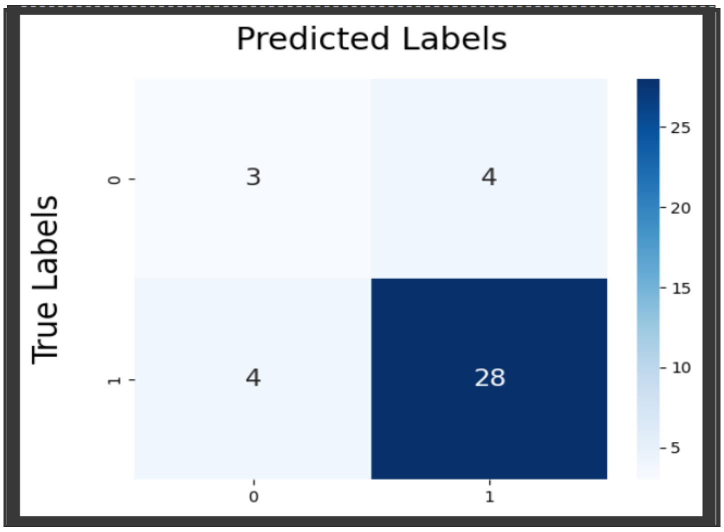


```
naive_bayes_classifier = GaussianNB()
naive_bayes_classifier.fit(X_train, y_train)


y_pred = naive_bayes_classifier.predict(X_test)
```

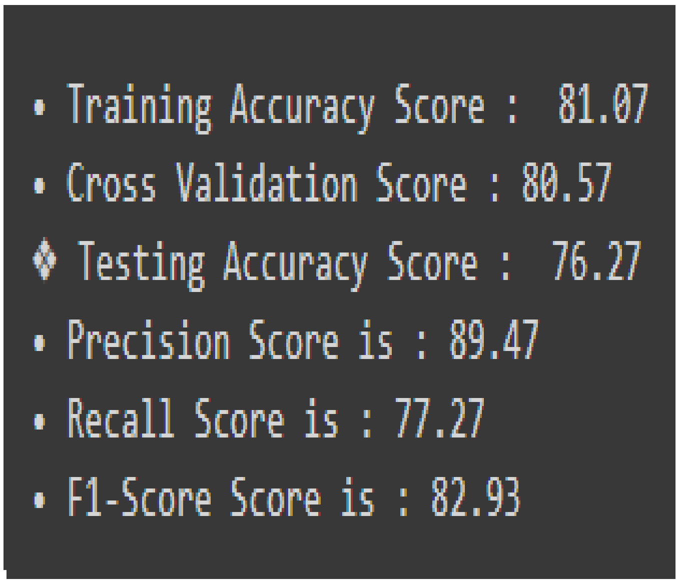
Classification Report using Naïve Bayes on Unbalanced Data



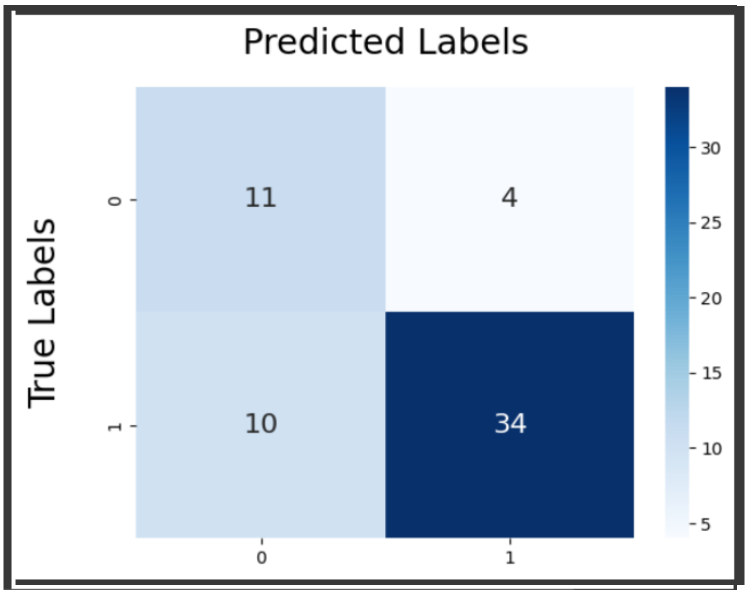
```
• Training Accuracy Score :  79.49
• Cross Validation Score : 75.62
❖ Testing Accuracy Score :  87.18
• Precision Score is : 90.91
• Recall Score is : 93.75
• F1-Score Score is : 92.31
-----------------------------------------
```

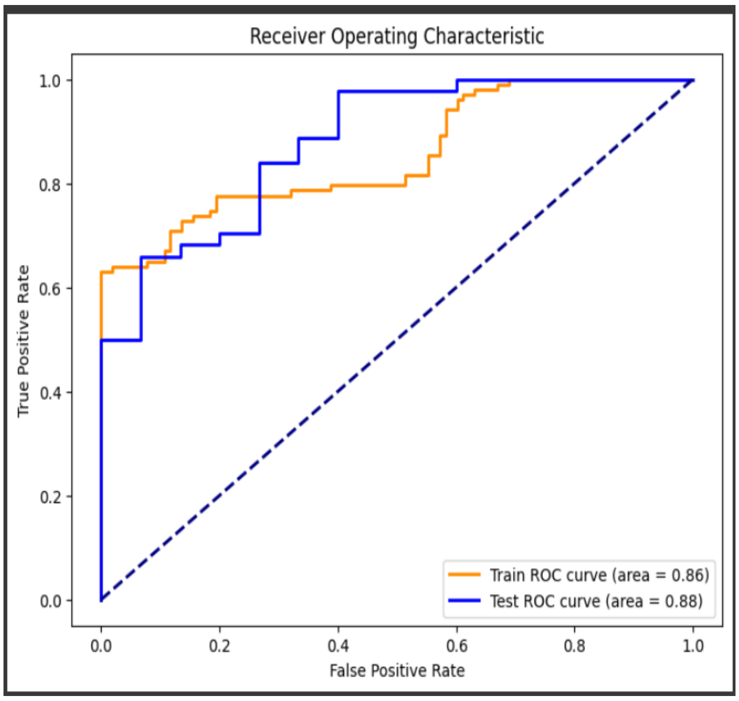


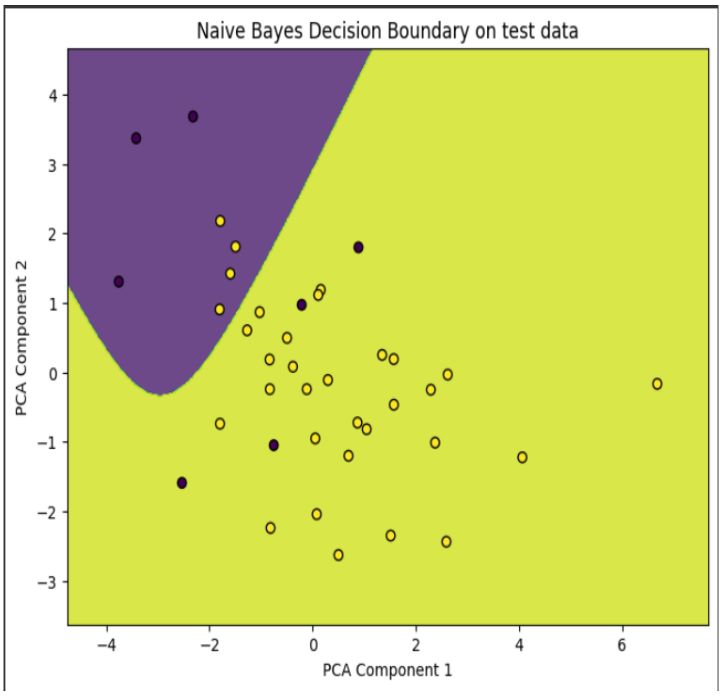
Balanced Dataset

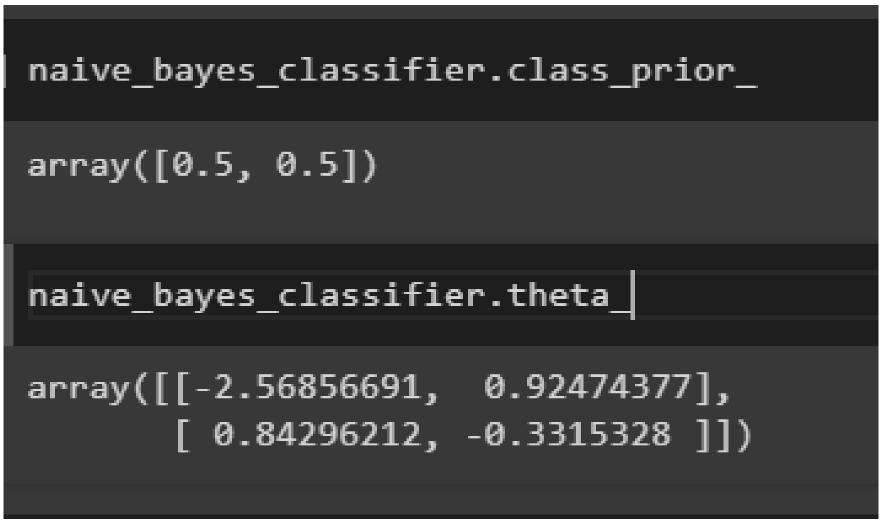


```
• Training Accuracy Score :  81.07
• Cross Validation Score : 80.57
• Testing Accuracy Score :  76.27
• Precision Score is : 89.47
• Recall Score is : 77.27
• F1-Score Score is : 82.93
```







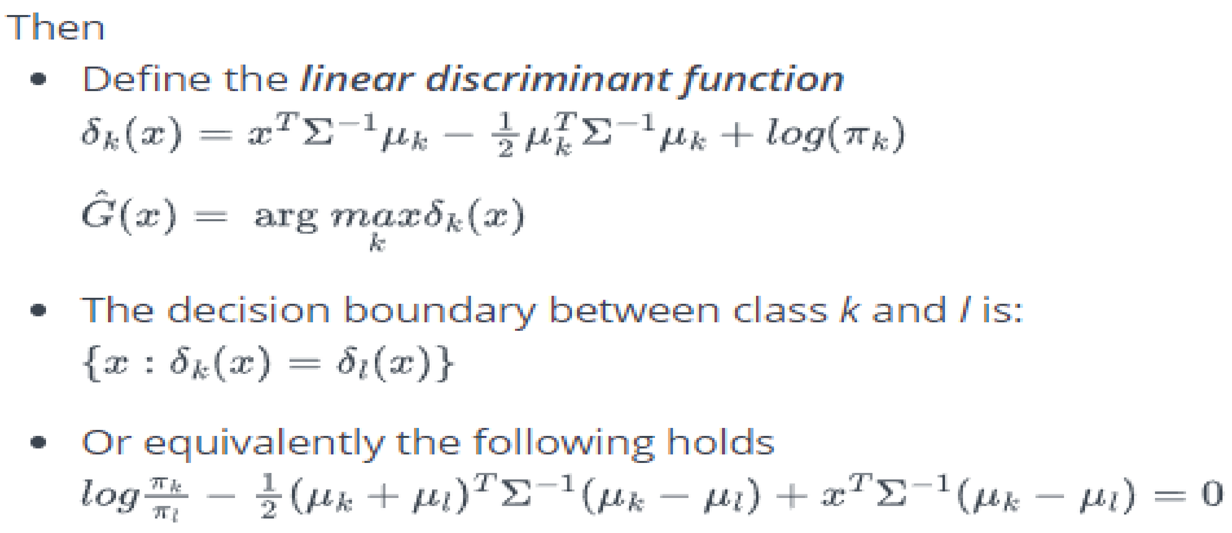


```
naive_bayes_classifier.class_prior_

array([0.5, 0.5])

naive_bayes_classifier.theta_

array([[-2.56856691,  0.92474377],
       [ 0.84296212, -0.3315328 ]])
```

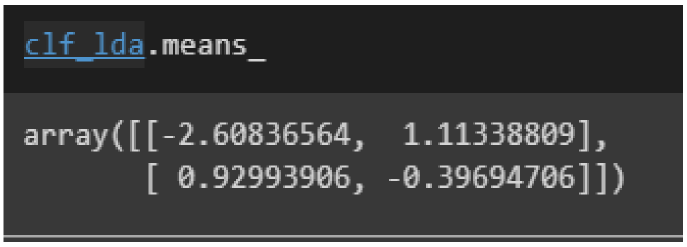
Linear Discriminant Analysis

Under LDA we assume that the density for X, given every class k is following a Gaussian distribution.

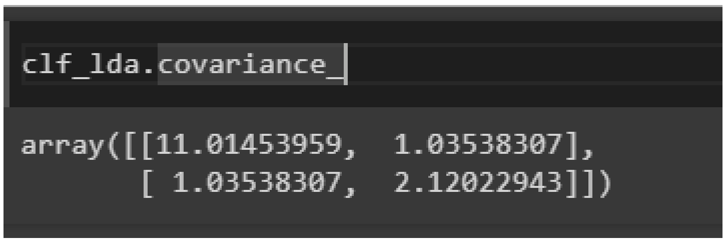
For Linear discriminant analysis (LDA) $\sum = \sum_K$. Mean and covariance of lda is given below


Then

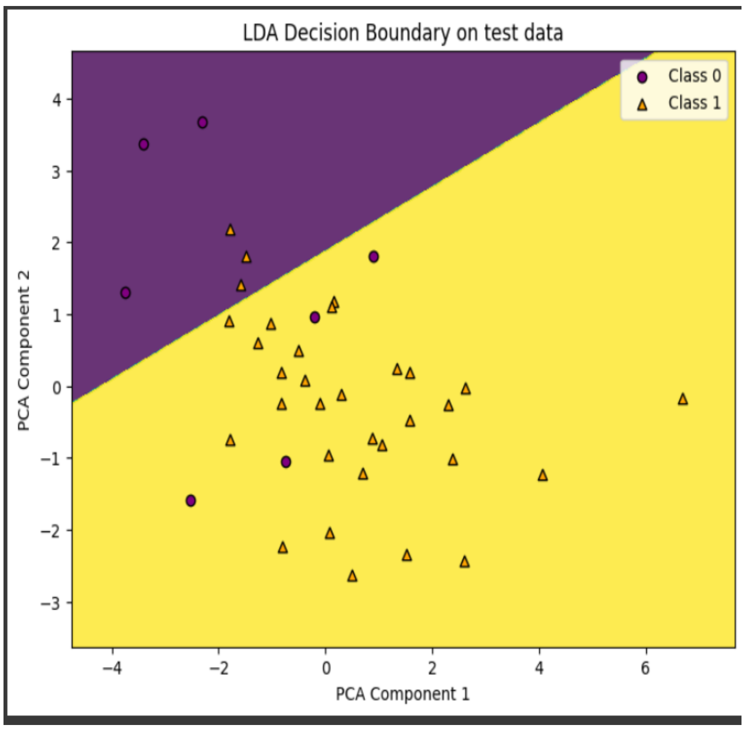
- Define the *linear discriminant function*
  $\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log(\pi_k)$
  $\hat{G}(x) = \arg\max_k \delta_k(x)$
- The decision boundary between class *k* and *l* is:
  $\{x : \delta_k(x) = \delta_l(x)\}$
- Or equivalently the following holds
  $log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T\Sigma^{-1}(\mu_k - \mu_l) + x^T\Sigma^{-1}(\mu_k - \mu_l) = 0$

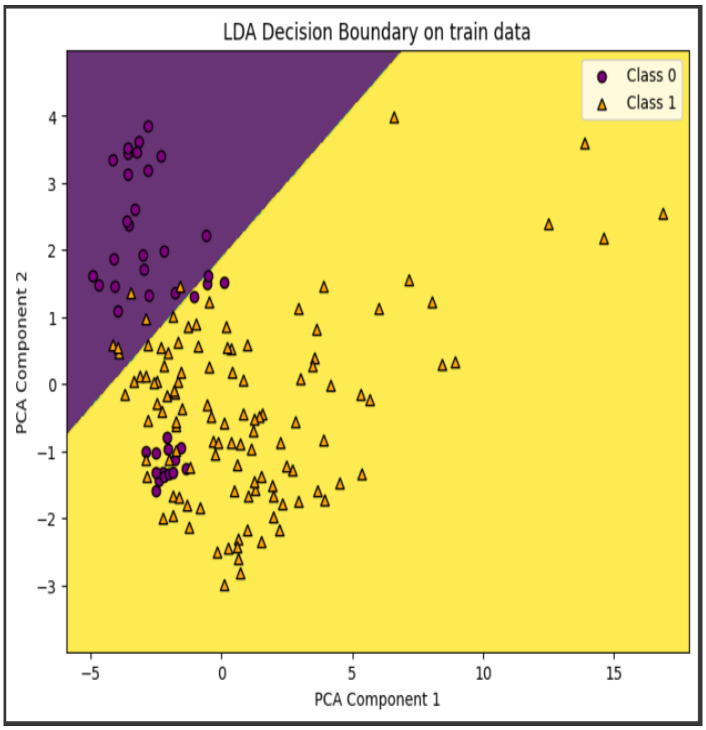

```
clf_lda.means_
```

```
array([[-2.60836564,  1.11338809],
       [ 0.92993906, -0.39694706]])
```

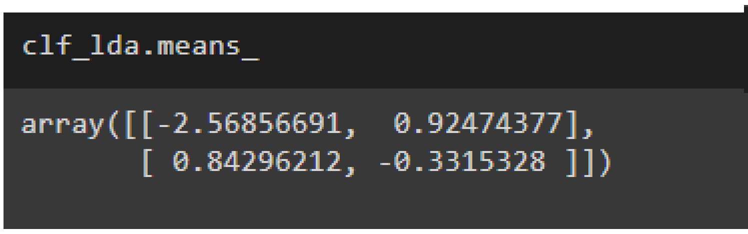

```
clf_lda.covariance_
```

```
array([[11.01453959,  1.03538307],
       [ 1.03538307,  2.12022943]])
```

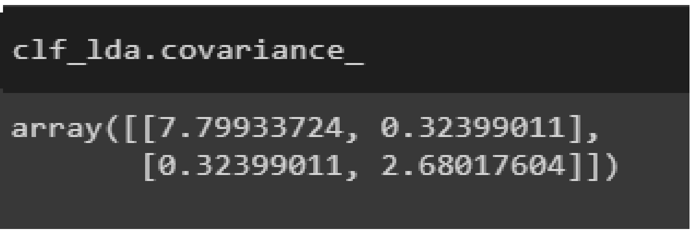




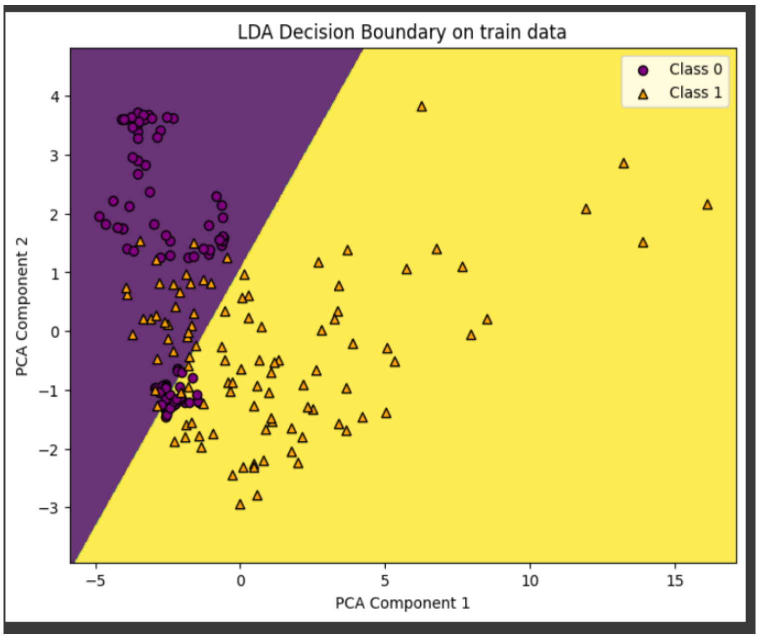
Balanced Data

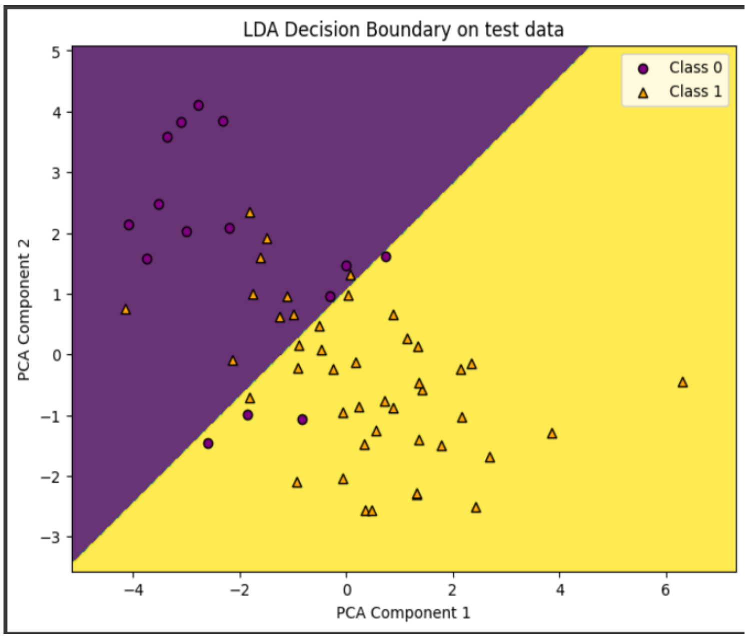


```
clf_lda.means_

array([[-2.56856691,  0.92474377],
       [ 0.84296212, -0.3315328 ]])
```

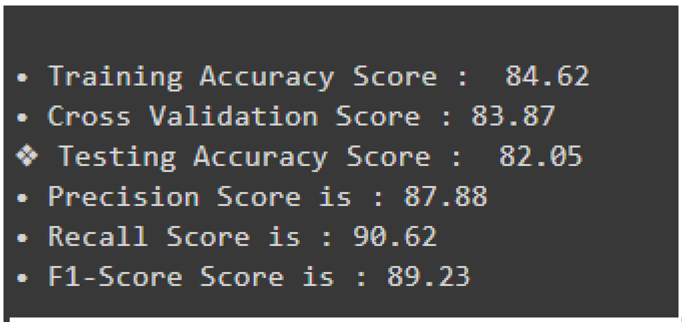


```
clf_lda.covariance_

array([[7.79933724, 0.32399011],
       [0.32399011, 2.68017604]])
```

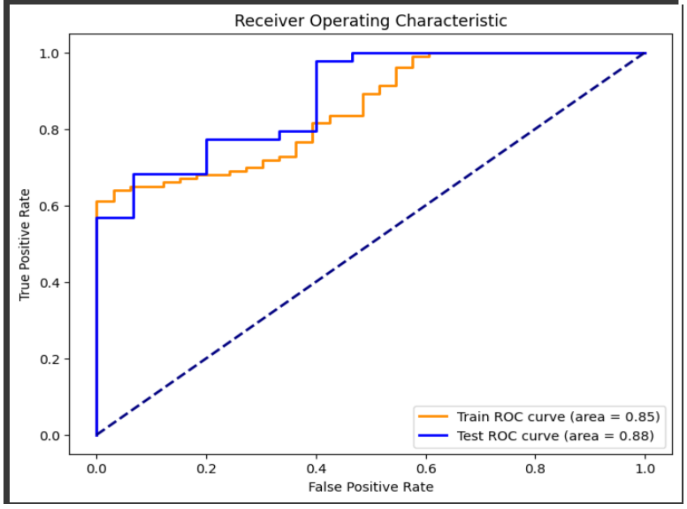


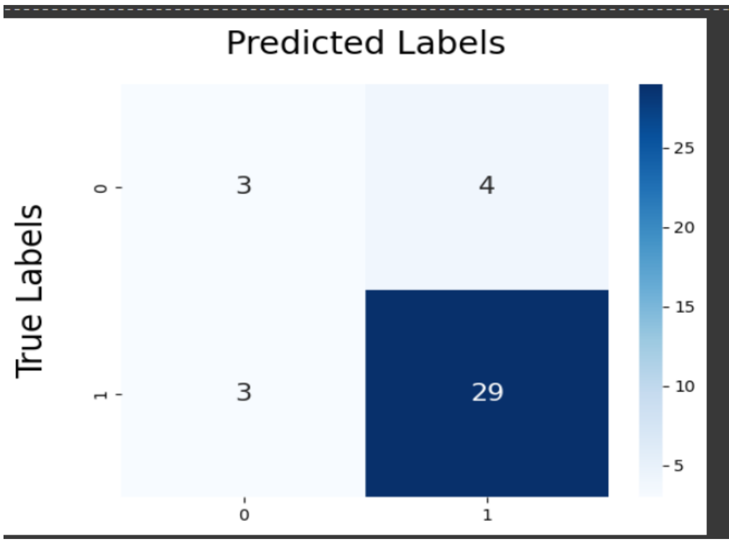


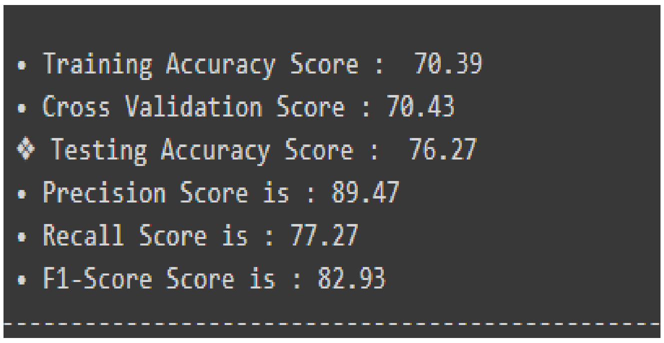
Classification Report of Unbalance vs Balanced



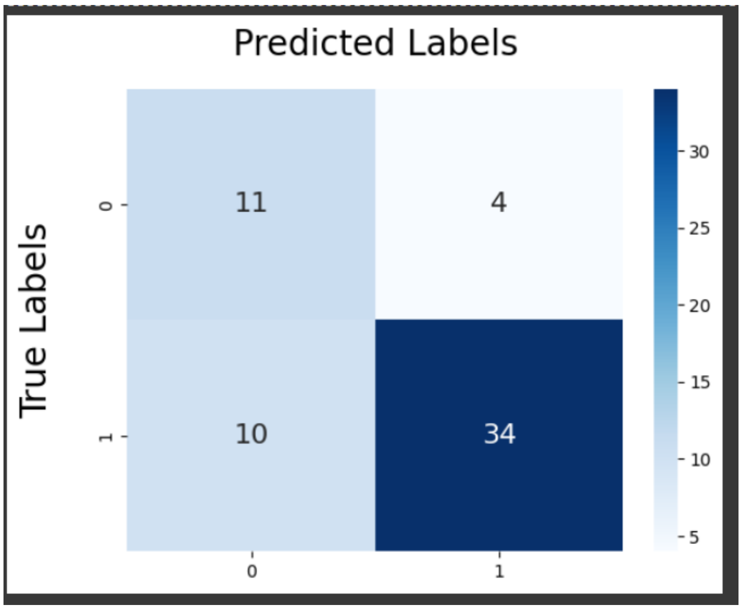
```
• Training Accuracy Score :  84.62
• Cross Validation Score : 83.87
❖ Testing Accuracy Score :  82.05
• Precision Score is : 87.88
• Recall Score is : 90.62
• F1-Score Score is : 89.23
```

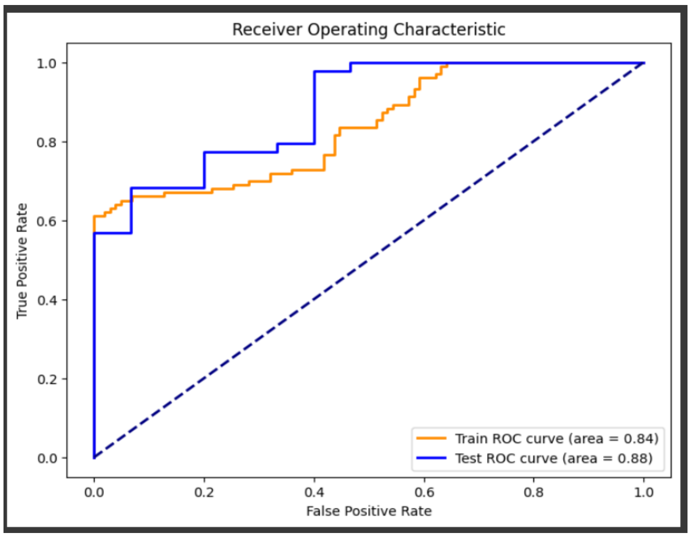






```
• Training Accuracy Score :  70.39
• Cross Validation Score : 70.43
❖ Testing Accuracy Score :  76.27
• Precision Score is : 89.47
• Recall Score is : 77.27
• F1-Score Score is : 82.93
-----------------------------------------------
```





Quadratic Discriminant Analysis

QDA is not really that much different from LDA except that you assume that the covariance matrix can be different for each class and so, we will estimate the covariance matrix separately for each class k, k =1, 2, ...

Quadratic discriminant function:

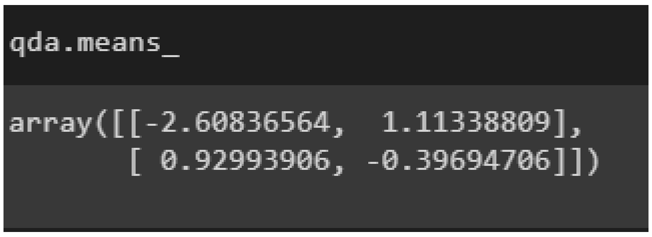
$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k$$

This quadratic discriminant function is very much like the linear discriminant function except that because Σk, the covariance matrix, is not identical, you cannot throw away the quadratic terms. This discriminant function is a quadratic function and will contain second order terms.

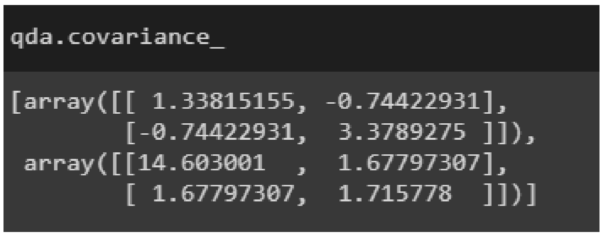
Classification rule:

$$\hat{G}(x) = \arg\max_k \delta_k(x)$$

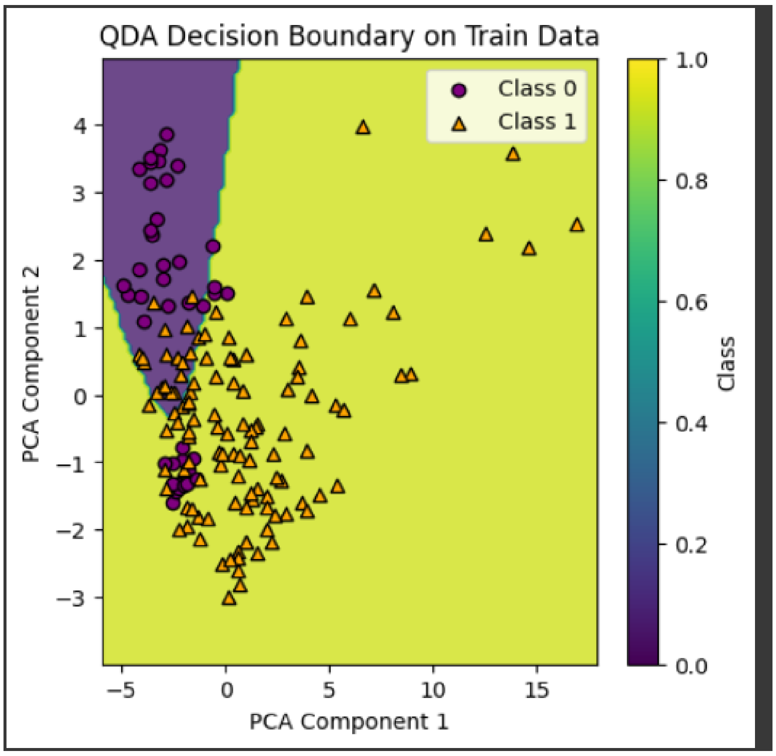
Unbalanced Data

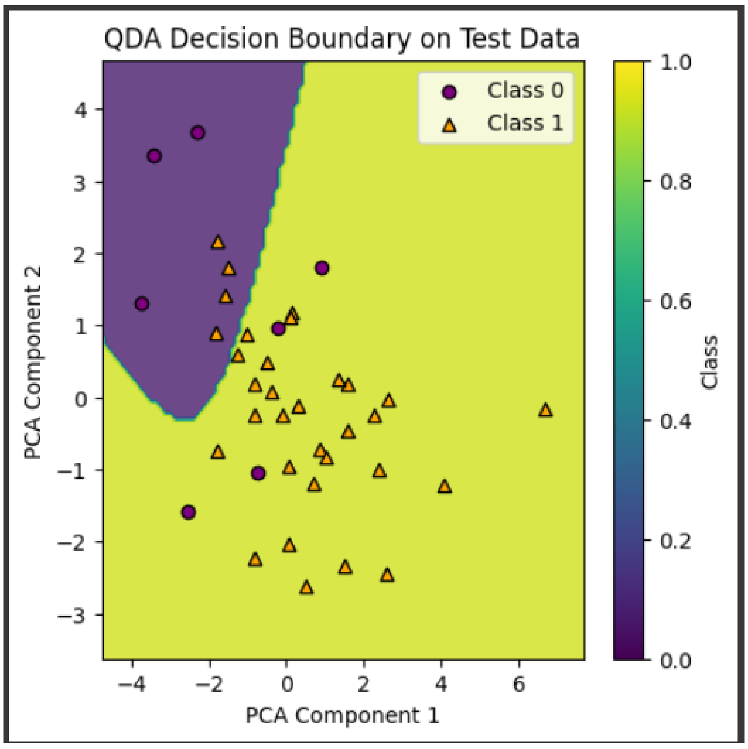


```
qda.means_

array([[-2.60836564,  1.11338809],
       [ 0.92993906, -0.39694706]])
```

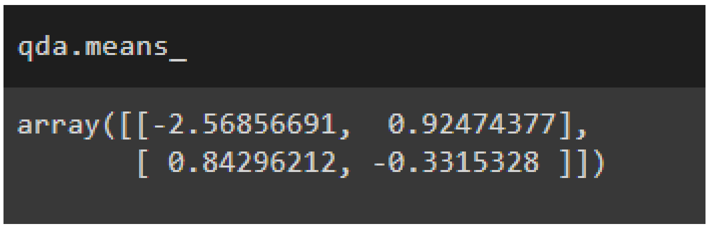


```
qda.covariance_

[array([[ 1.33815155, -0.74422931],
        [-0.74422931,  3.3789275 ]]),
 array([[14.603001  ,  1.67797307],
        [ 1.67797307,  1.715778  ]])]
```



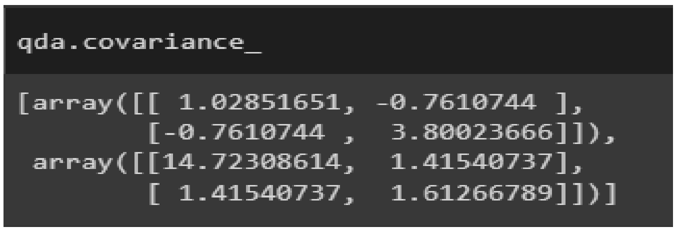


Balanced Data



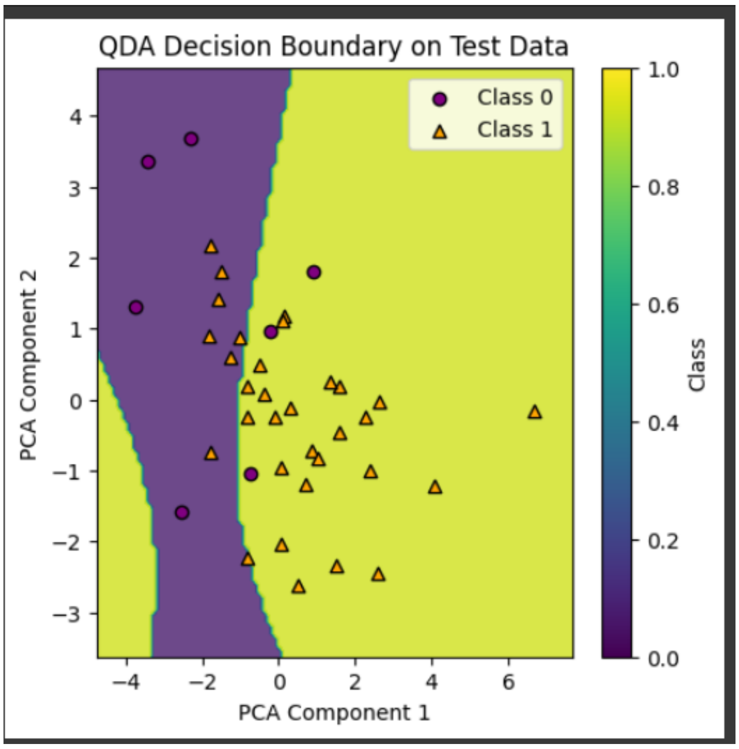
```
qda.means_
```

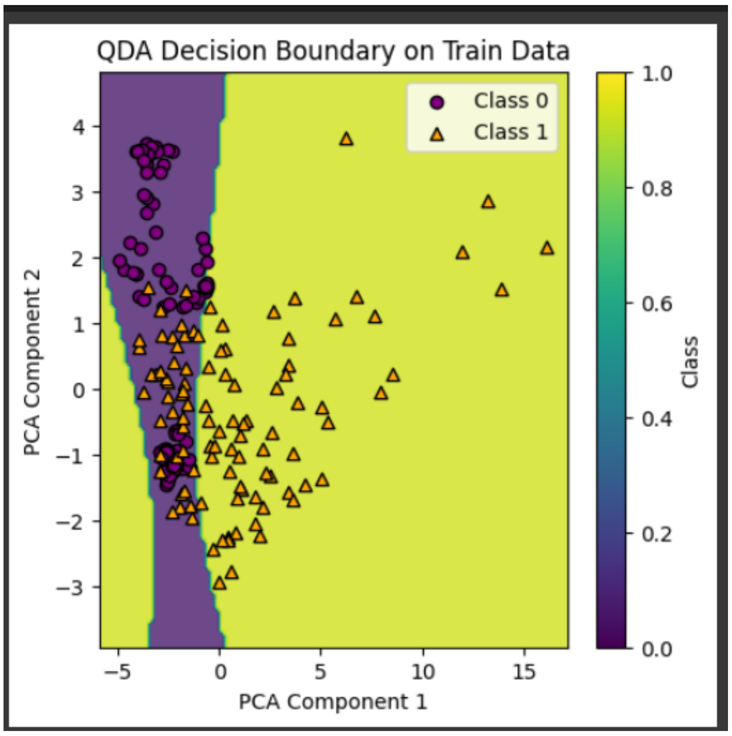
```
array([[-2.56856691,  0.92474377],
       [ 0.84296212, -0.3315328 ]])
```



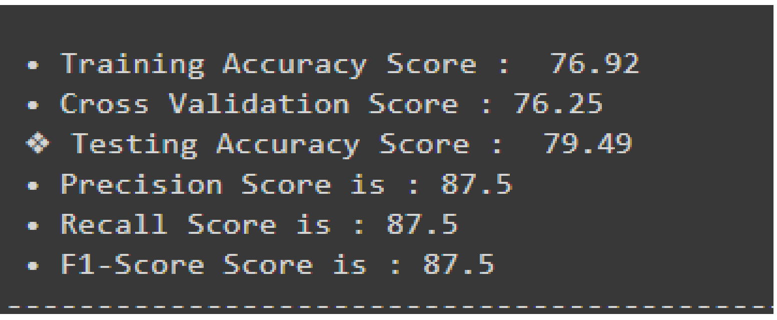
```
qda.covariance_
```

```
[array([[ 1.02851651, -0.7610744 ],
        [-0.7610744 ,  3.80023666]]),
 array([[14.72308614,  1.41540737],
        [ 1.41540737,  1.61266789]])]
```

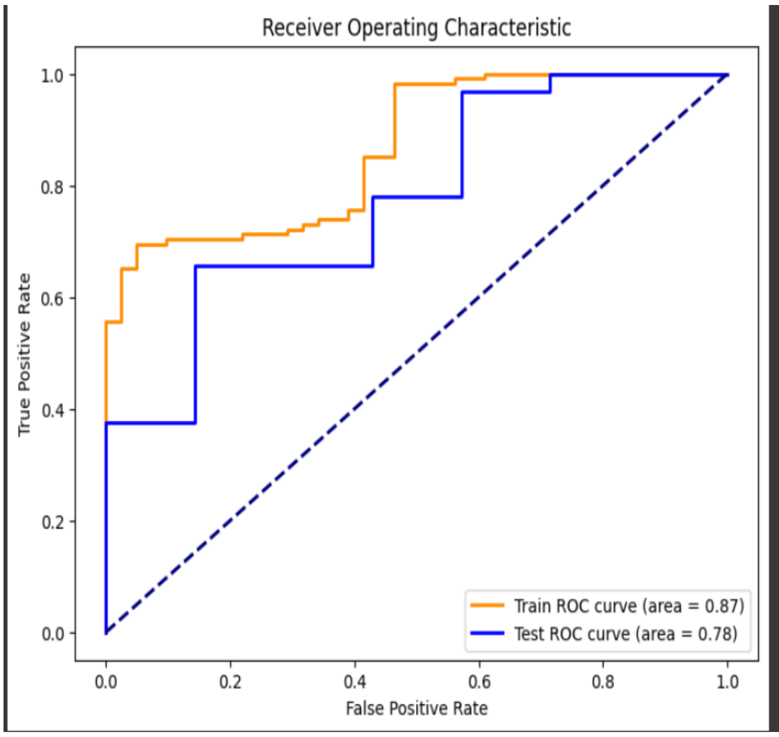


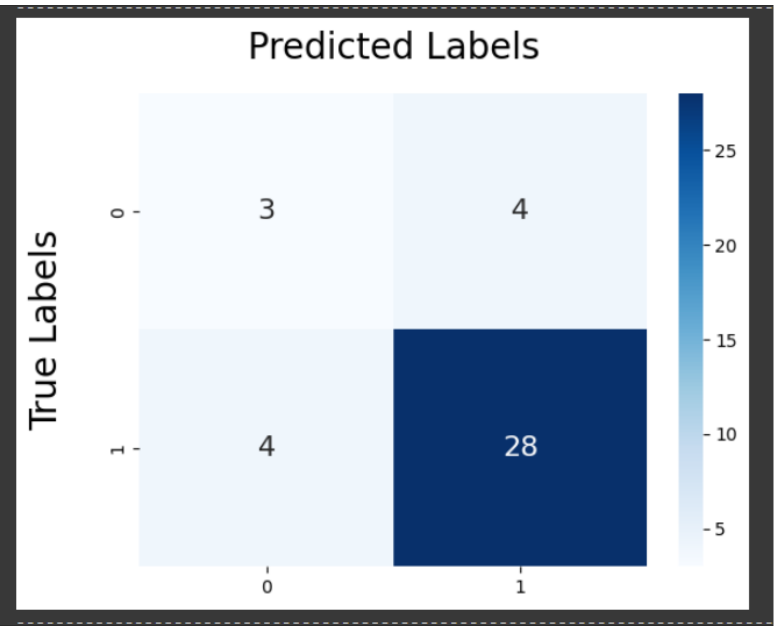


Classification Report of Unbalanced Data vs Balanced Data

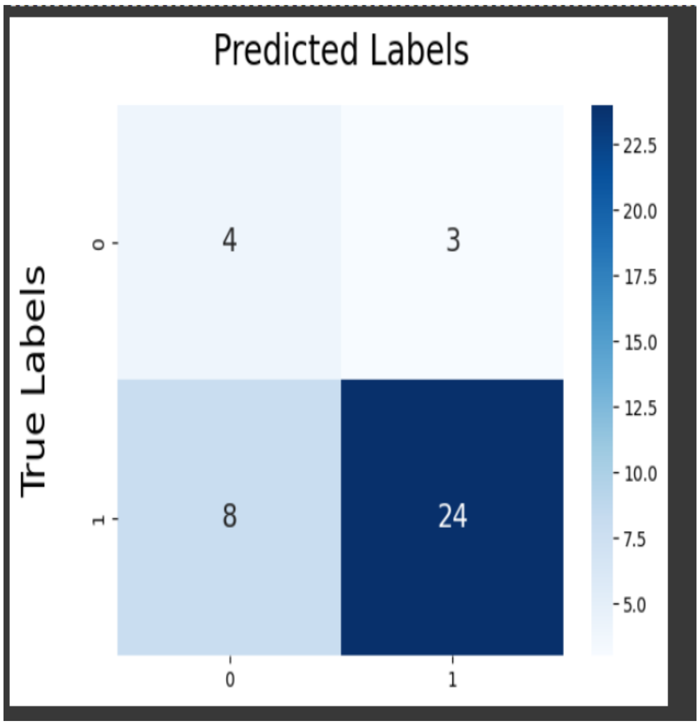


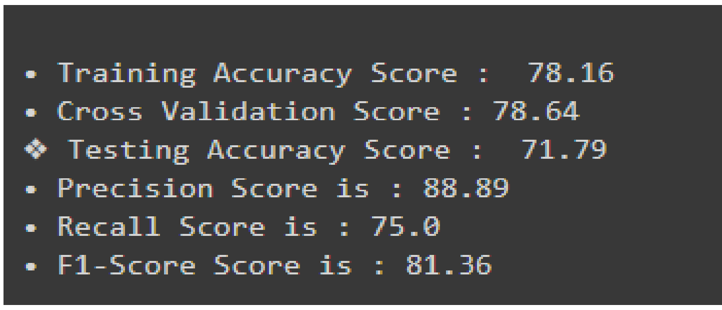
```
• Training Accuracy Score :  76.92
• Cross Validation Score : 76.25
❖ Testing Accuracy Score :  79.49
• Precision Score is : 87.5
• Recall Score is : 87.5
• F1-Score Score is : 87.5
```



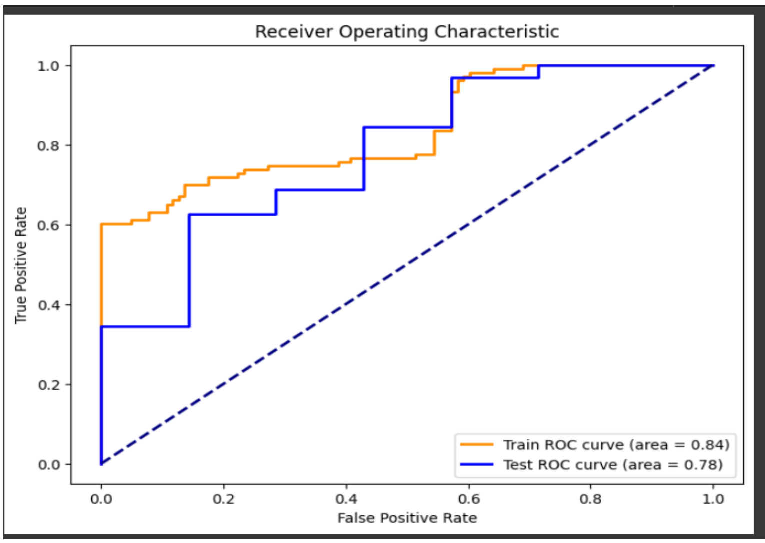


Balanced Data

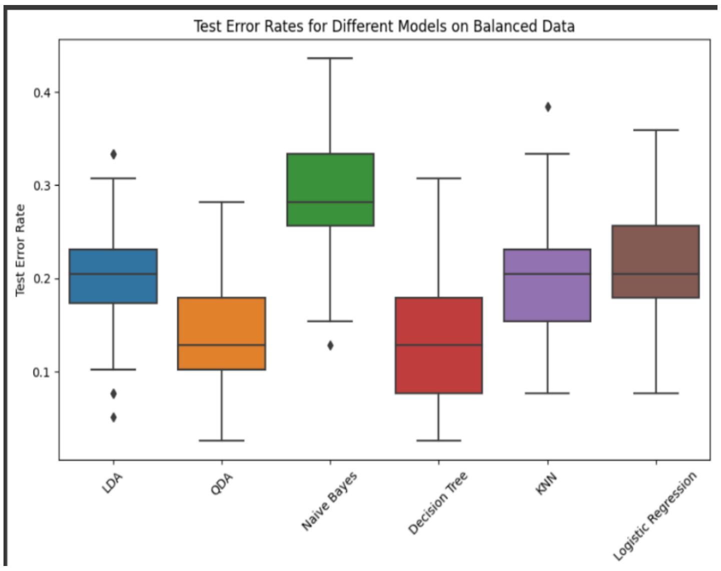


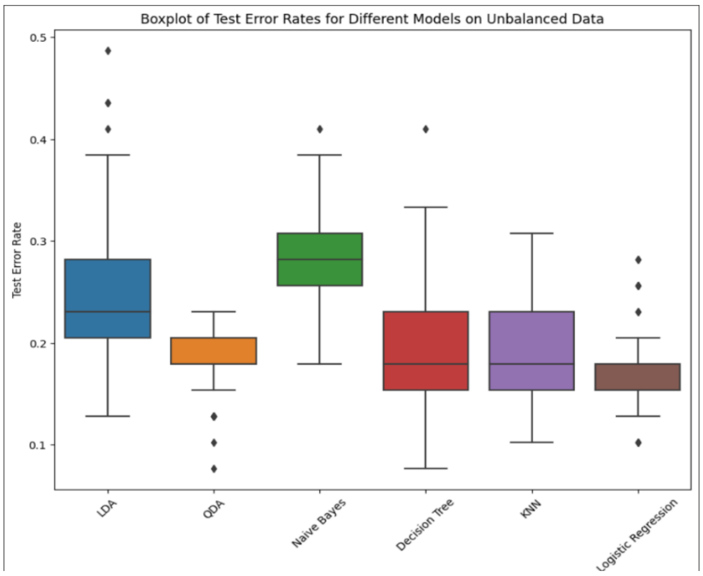


```
• Training Accuracy Score :  78.16
• Cross Validation Score : 78.64
❖ Testing Accuracy Score :  71.79
• Precision Score is : 88.89
• Recall Score is : 75.0
• F1-Score Score is : 81.36
```



Analytical Comparison





The left-hand panel shows that QDA and decision tree has outperformed all of the other approaches. LDA has performed worse than QDA, since it fit a less flexible classifier. The notable exception is naive Bayes, which performs very poorly here, since it may be possible that Naive Bayes assumption of independent predictors is violated.

Conclusion:

- Applying the oversampling technique to balance the class distribution in our dataset, and found that this approach improved the performance of our machine learning models. Specifically, the balancing technique helped to reduce the bias towards the majority class and improve the accuracy of the minority class predictions, resulting in a more balanced and accurate model overall.

- While achieving 100% accuracy on both the training and testing sets may seem like an ideal scenario, it is not always the best indicator of model performance. In fact, achieving perfect accuracy on both sets can sometimes be a sign of overfitting, where the model has simply memorized the training data and may not generalize well to new, unseen data. In contrast, a model with high accuracy on the training set but slightly lower accuracy on the testing set may indicate better generalization to new data. This is because the model has learned to identify patterns in the training data that are also present in the testing data, rather than simply memorizing the training set.

- Based on our analysis and evaluation, we recommend using the Decision Tree or QDA model over the other model for this particular problem.