Project Development
Phase Model
Performance Test

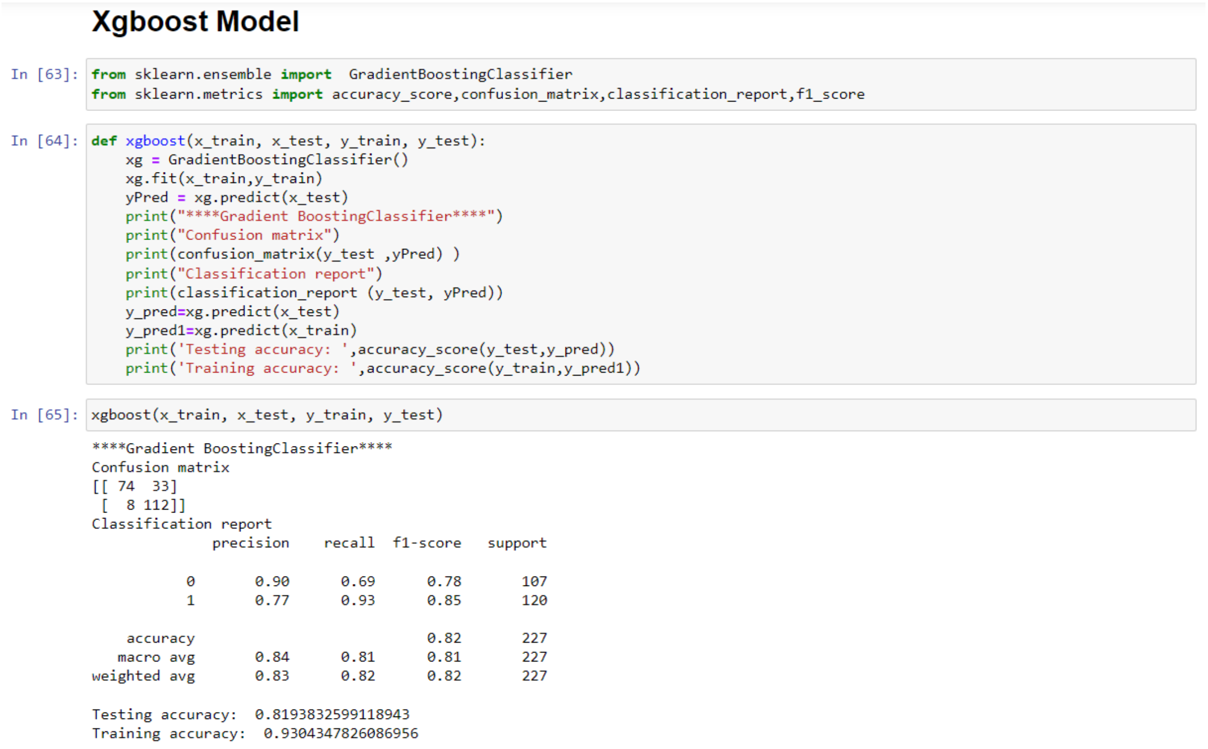
Date	10 November 2022
Team ID	PNT2022TMID35767
Project Name	Project - Smart Lender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	10 Marks

Model Performance Testing:

For our model performance testing, we are using XG-boost for prediction.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - , MSE - , RMSE - , R2 score - **Classification Model:** Confusion Matrix - , Accuracy Score- & Classification Report -	FIGURE-1
2.	Tune the Model	Hyper parameter Tuning - Validation Method -	FIGURE-2

FIGURE – 1



Xgboost Model

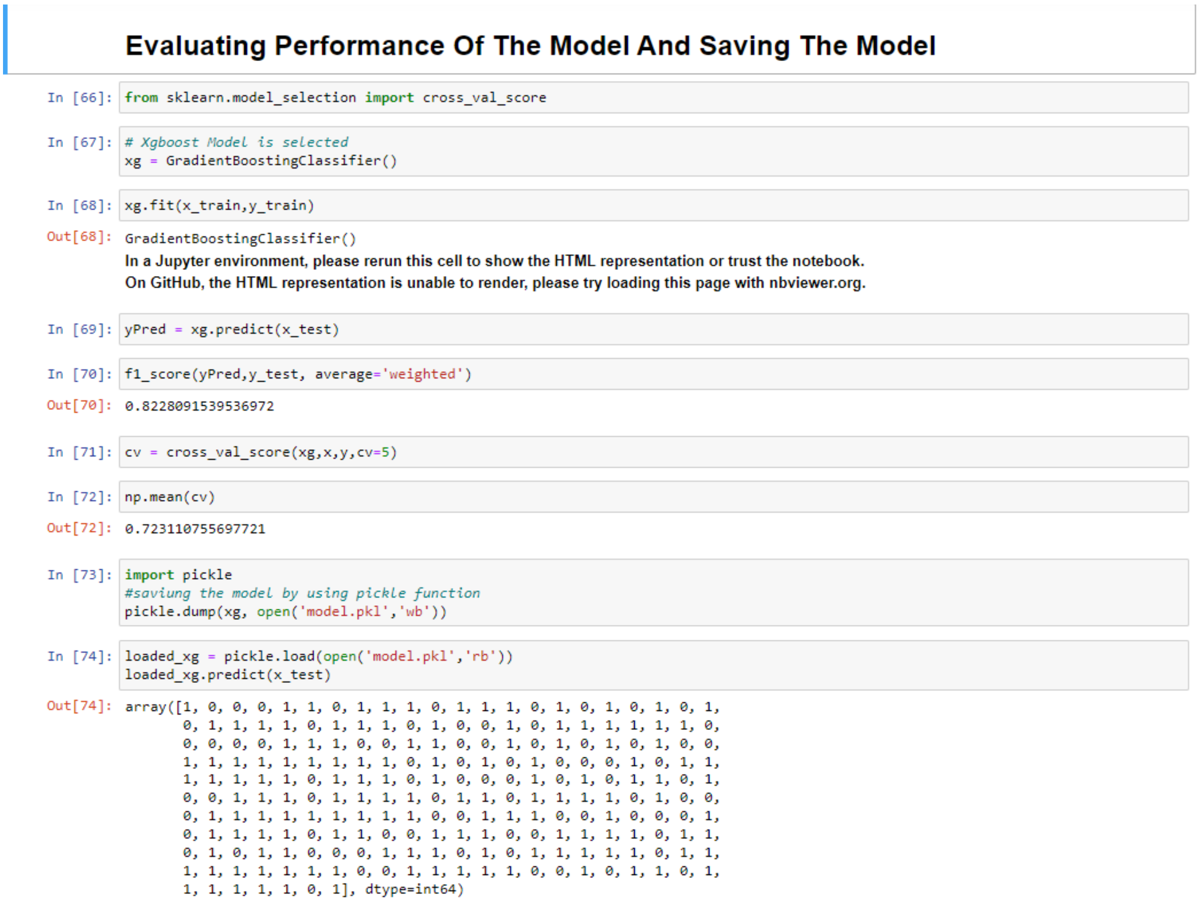
In [63]:
```
from sklearn.ensemble import  GradientBoostingClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score
```

In [64]:
```
def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print("****Gradient BoostingClassifier****")
    print("Confusion matrix")
    print(confusion_matrix(y_test ,yPred) )
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=xg.predict(x_test)
    y_pred1=xg.predict(x_train)
    print('Testing accuracy: ',accuracy_score(y_test,y_pred))
    print('Training accuracy: ',accuracy_score(y_train,y_pred1))
```

In [65]:
```
xgboost(x_train, x_test, y_train, y_test)
```

```
****Gradient BoostingClassifier****
Confusion matrix
[[ 74  33]
 [  8 112]]
Classification report
              precision    recall  f1-score   support

           0       0.90      0.69      0.78       107
           1       0.77      0.93      0.85       120

    accuracy                           0.82       227
   macro avg       0.84      0.81      0.81       227
weighted avg       0.83      0.82      0.82       227

Testing accuracy:  0.8193832599118943
Training accuracy:  0.9304347826086956
```

FIGURE – 2

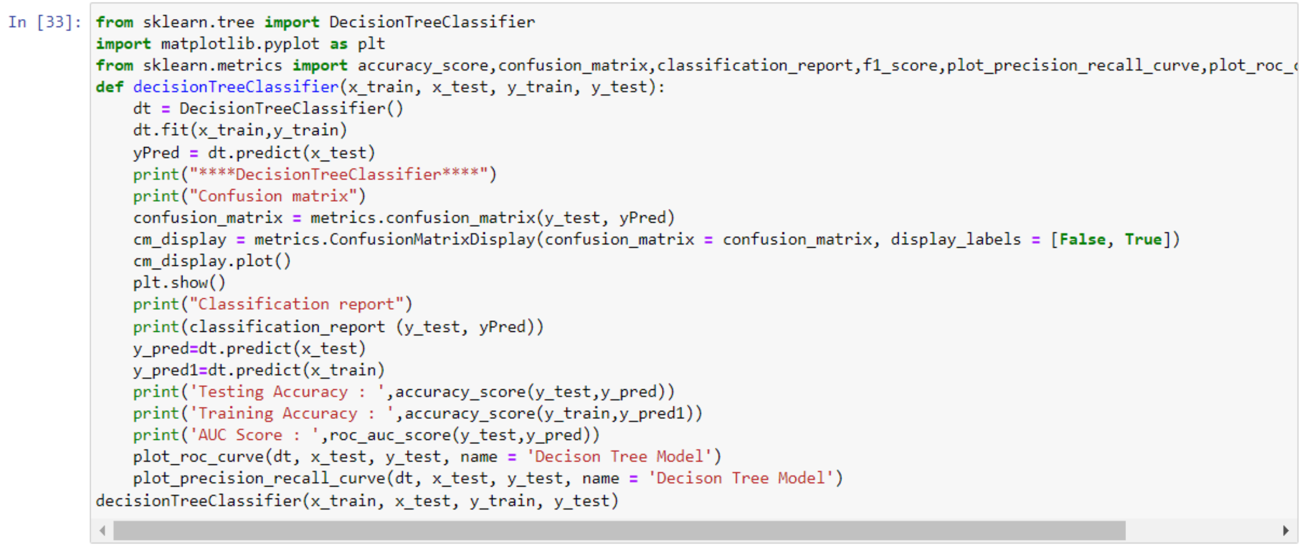


Evaluating Performance Of The Model And Saving The Model

```
In [66]: from sklearn.model_selection import cross_val_score

In [67]: # Xgboost Model is selected
         xg = GradientBoostingClassifier()

In [68]: xg.fit(x_train,y_train)

Out[68]: GradientBoostingClassifier()
         In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
         On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [69]: yPred = xg.predict(x_test)

In [70]: f1_score(yPred,y_test, average='weighted')

Out[70]: 0.8228091539536972

In [71]: cv = cross_val_score(xg,x,y,cv=5)

In [72]: np.mean(cv)

Out[72]: 0.723110755697721

In [73]: import pickle
         #saviung the model by using pickle function
         pickle.dump(xg, open('model.pkl','wb'))

In [74]: loaded_xg = pickle.load(open('model.pkl','rb'))
         loaded_xg.predict(x_test)

Out[74]: array([1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1,
                0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0,
                0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1,
                1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1,
                0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
                0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
                0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1,
                0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1,
                1, 1, 1, 1, 1, 0, 1], dtype=int64)
```

DECISION TREE CLASSIFIER

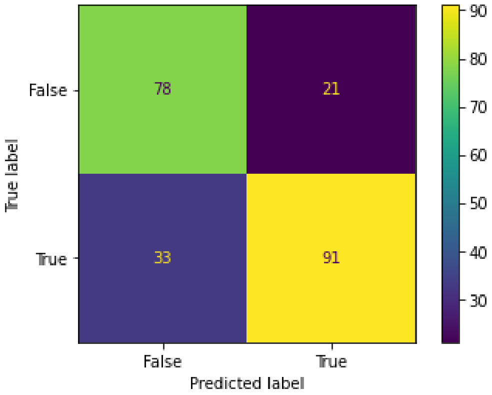
SOURCE CODE



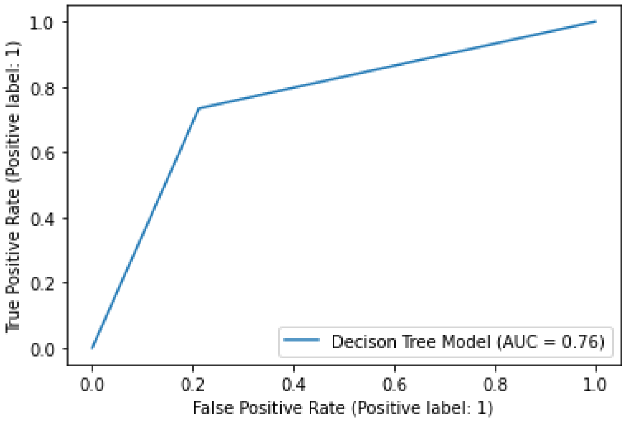
```
In [33]: from sklearn.tree import DecisionTreeClassifier
         import matplotlib.pyplot as plt
         from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score,plot_precision_recall_curve,plot_roc_c
         def decisionTreeClassifier(x_train, x_test, y_train, y_test):
             dt = DecisionTreeClassifier()
             dt.fit(x_train,y_train)
             yPred = dt.predict(x_test)
             print("****DecisionTreeClassifier****")
             print("Confusion matrix")
             confusion_matrix = metrics.confusion_matrix(y_test, yPred)
             cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
             cm_display.plot()
             plt.show()
             print("Classification report")
             print(classification_report (y_test, yPred))
             y_pred=dt.predict(x_test)
             y_pred1=dt.predict(x_train)
             print('Testing Accuracy : ',accuracy_score(y_test,y_pred))
             print('Training Accuracy : ',accuracy_score(y_train,y_pred1))
             print('AUC Score : ',roc_auc_score(y_test,y_pred))
             plot_roc_curve(dt, x_test, y_test, name = 'Decison Tree Model')
             plot_precision_recall_curve(dt, x_test, y_test, name = 'Decison Tree Model')
         decisionTreeClassifier(x_train, x_test, y_train, y_test)
```

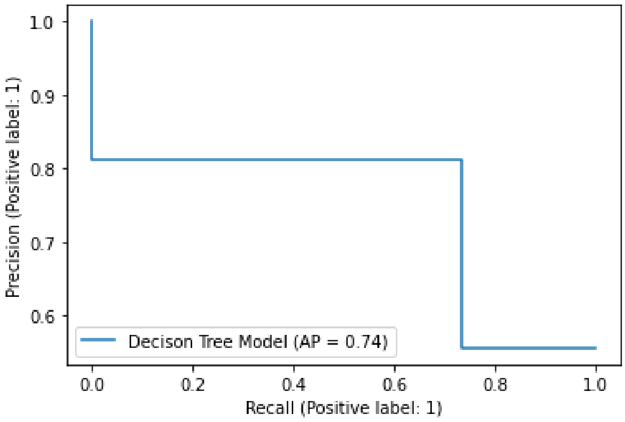
OUTPUT

```
****DecisionTreeClassifier****
Confusion matrix
```



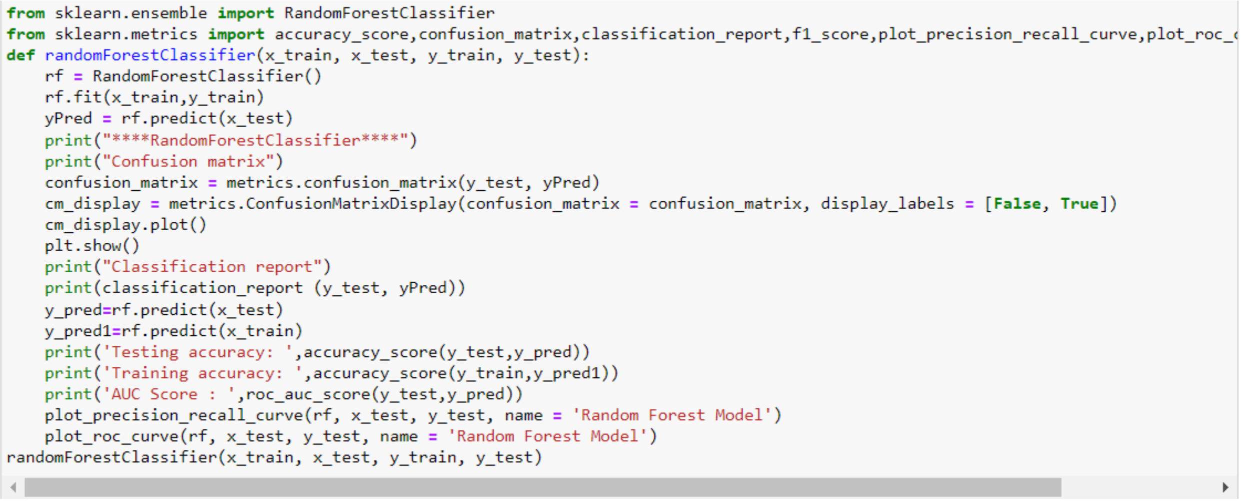
```
Classification report
              precision    recall  f1-score   support

           0       0.70      0.79      0.74        99
           1       0.81      0.73      0.77       124

    accuracy                           0.76       223
   macro avg       0.76      0.76      0.76       223
weighted avg       0.76      0.76      0.76       223

Testing Accuracy :  0.757847533632287
Training Accuracy :  1.0
AUC Score :  0.7608748778103617
```





RANDOM FOREST CLASSIFIER

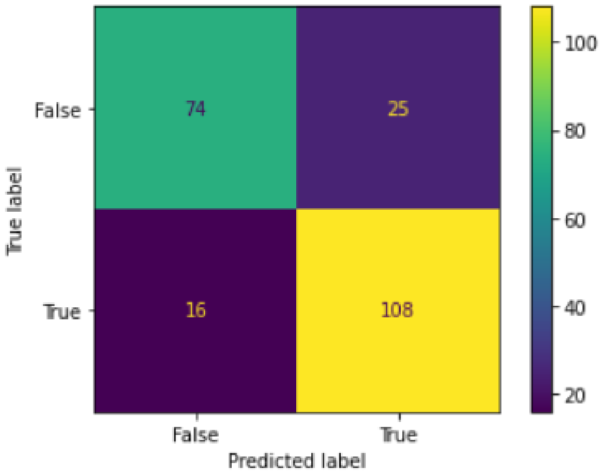
SOURCE CODE



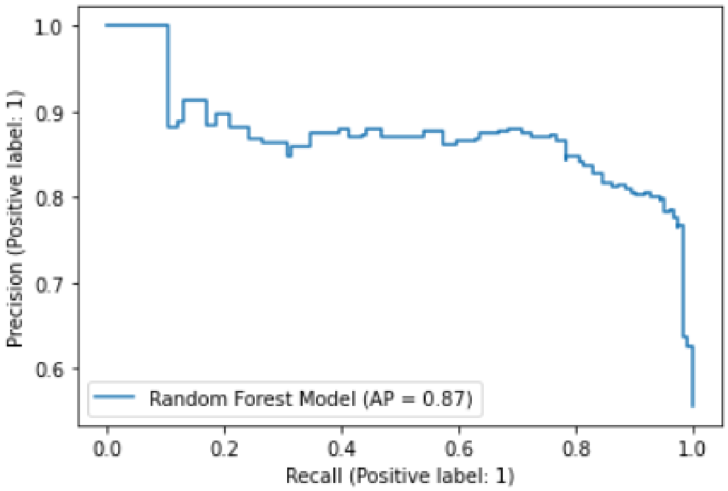
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score,plot_precision_recall_curve,plot_roc_c
def randomForestClassifier(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print("****RandomForestClassifier****")
    print("Confusion matrix")
    confusion_matrix = metrics.confusion_matrix(y_test, yPred)
    cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
    cm_display.plot()
    plt.show()
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=rf.predict(x_test)
    y_pred1=rf.predict(x_train)
    print('Testing accuracy: ',accuracy_score(y_test,y_pred))
    print('Training accuracy: ',accuracy_score(y_train,y_pred1))
    print('AUC Score : ',roc_auc_score(y_test,y_pred))
    plot_precision_recall_curve(rf, x_test, y_test, name = 'Random Forest Model')
    plot_roc_curve(rf, x_test, y_test, name = 'Random Forest Model')
randomForestClassifier(x_train, x_test, y_train, y_test)
```

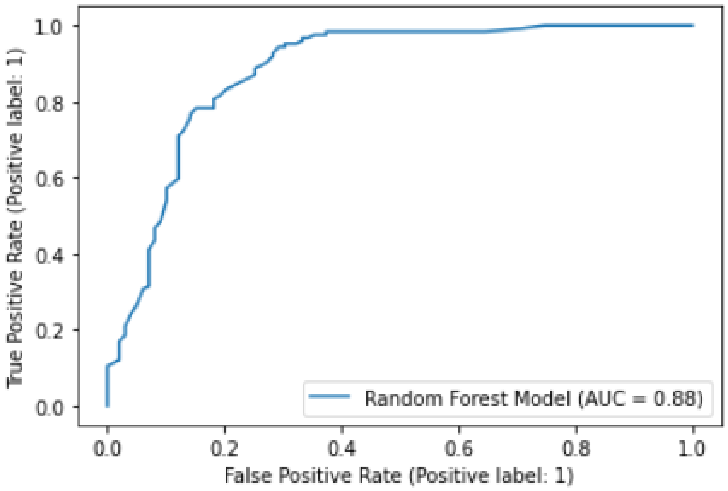
OUTPUT

```
****RandomForestClassifier****
Confusion matrix
```



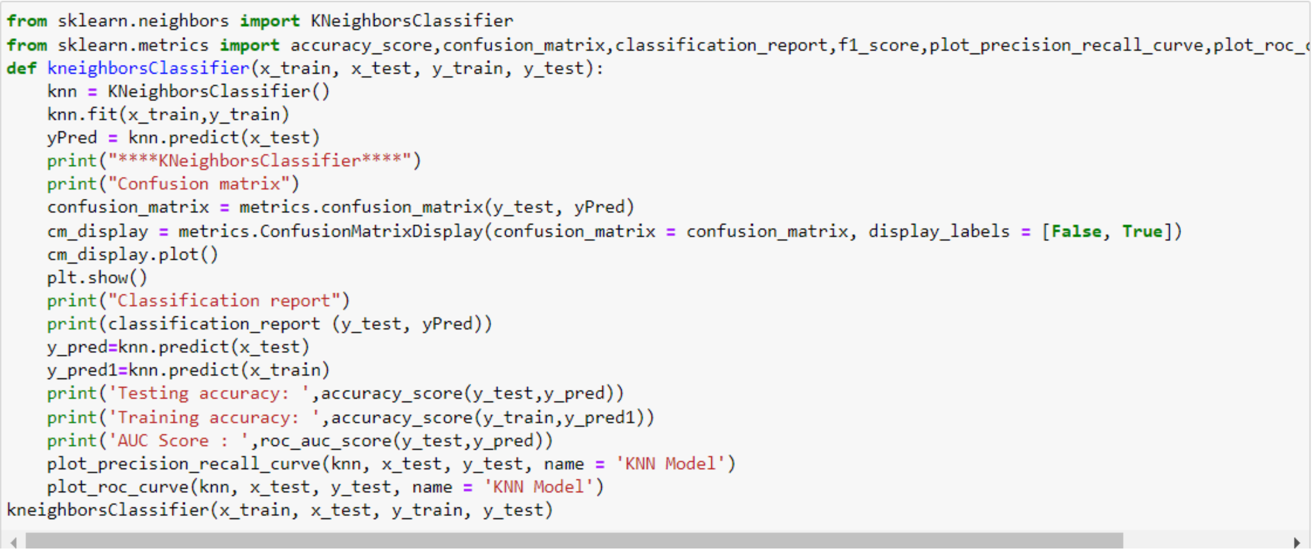
```
Classification report
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        99
           1       0.81      0.87      0.84       124

    accuracy                           0.82       223
   macro avg       0.82      0.81      0.81       223
weighted avg       0.82      0.82      0.81       223

Testing accuracy:  0.8161434977578476
Training accuracy:  1.0
AUC Score :  0.809221244705115 7
```





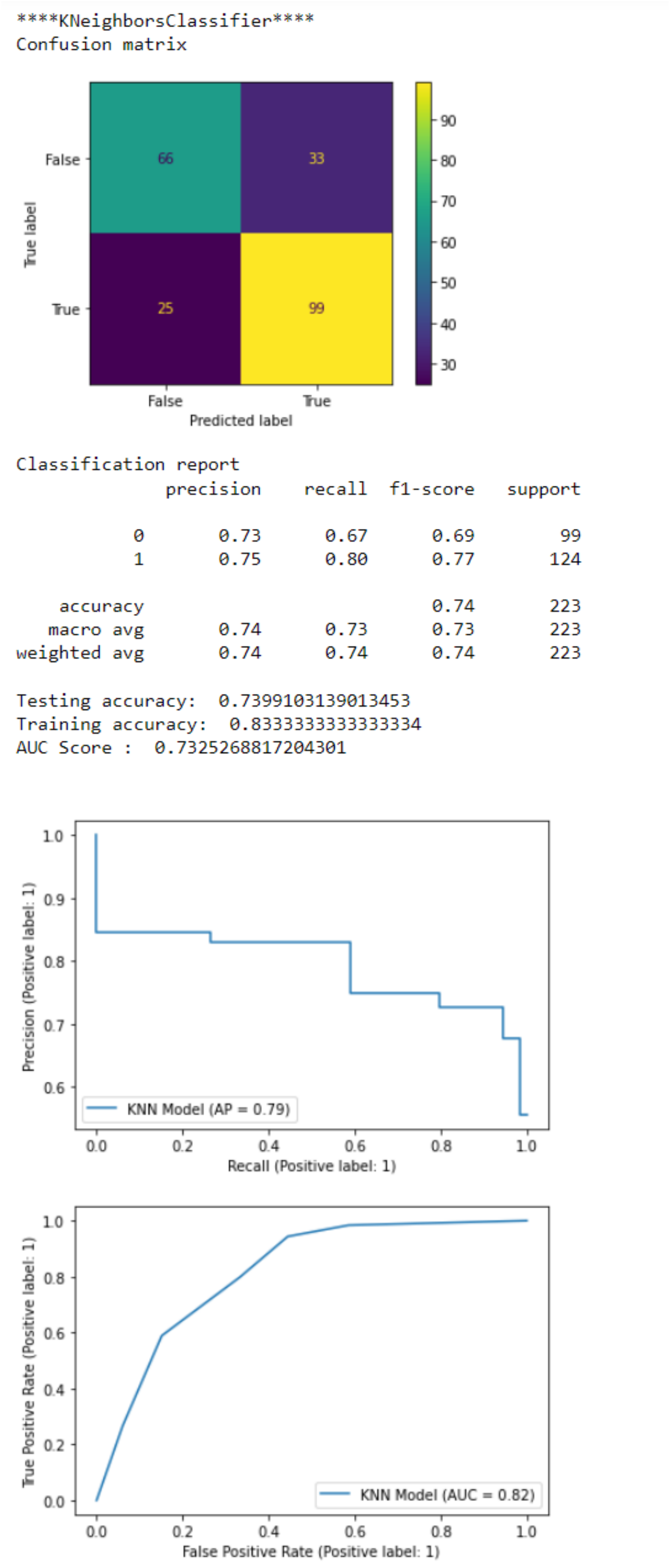
KNN CLASSIFIER

SOURCE CODE



```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score,plot_precision_recall_curve,plot_roc_c
def kneighborsClassifier(x_train, x_test, y_train, y_test):
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    yPred = knn.predict(x_test)
    print("****KNeighborsClassifier****")
    print("Confusion matrix")
    confusion_matrix = metrics.confusion_matrix(y_test, yPred)
    cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
    cm_display.plot()
    plt.show()
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=knn.predict(x_test)
    y_pred1=knn.predict(x_train)
    print('Testing accuracy: ',accuracy_score(y_test,y_pred))
    print('Training accuracy: ',accuracy_score(y_train,y_pred1))
    print('AUC Score : ',roc_auc_score(y_test,y_pred))
    plot_precision_recall_curve(knn, x_test, y_test, name = 'KNN Model')
    plot_roc_curve(knn, x_test, y_test, name = 'KNN Model')
kneighborsClassifier(x_train, x_test, y_train, y_test)
```

OUTPUT



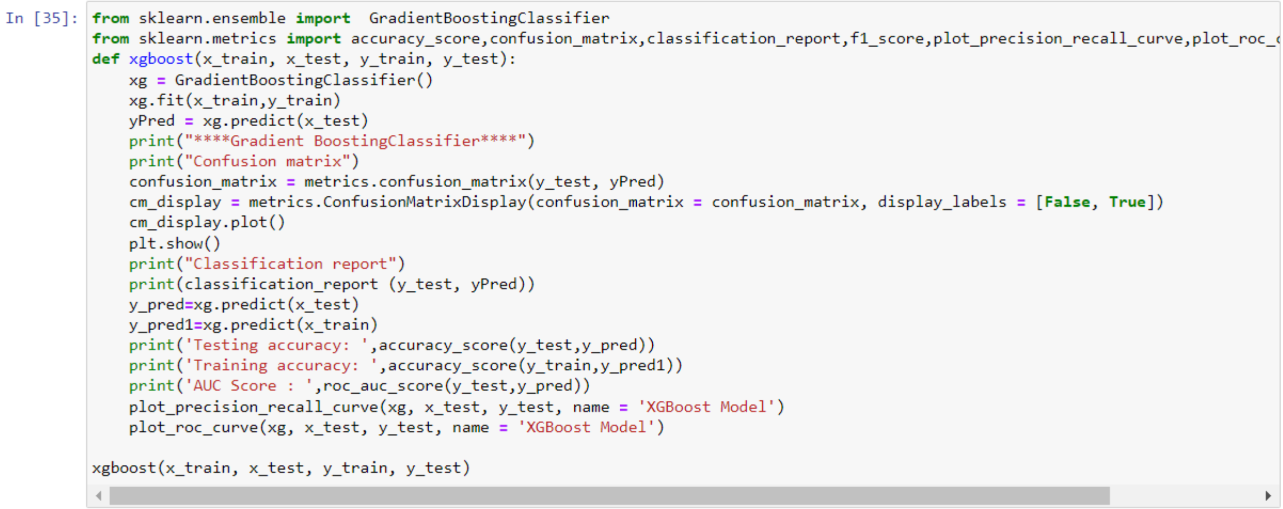
```
****KNeighborsClassifier****
Confusion matrix
```

```
Classification report
              precision    recall  f1-score   support

           0       0.73      0.67      0.69        99
           1       0.75      0.80      0.77       124

    accuracy                           0.74       223
   macro avg       0.74      0.73      0.73       223
weighted avg       0.74      0.74      0.74       223

Testing accuracy:  0.7399103139013453
Training accuracy:  0.8333333333333334
AUC Score :  0.7325268817204301
```

Precision (Positive label: 1)
1.0
0.9
0.8
0.7
0.6
KNN Model (AP = 0.79)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)
True Positive Rate (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
KNN Model (AUC = 0.82)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

GRADIENT BOOSTING CLASSIFIER

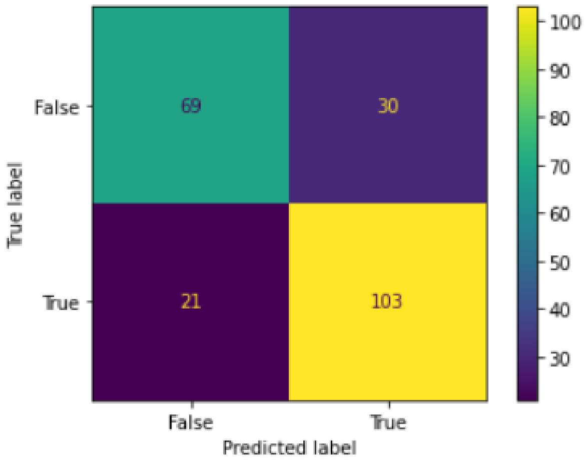
SOURCE CODE



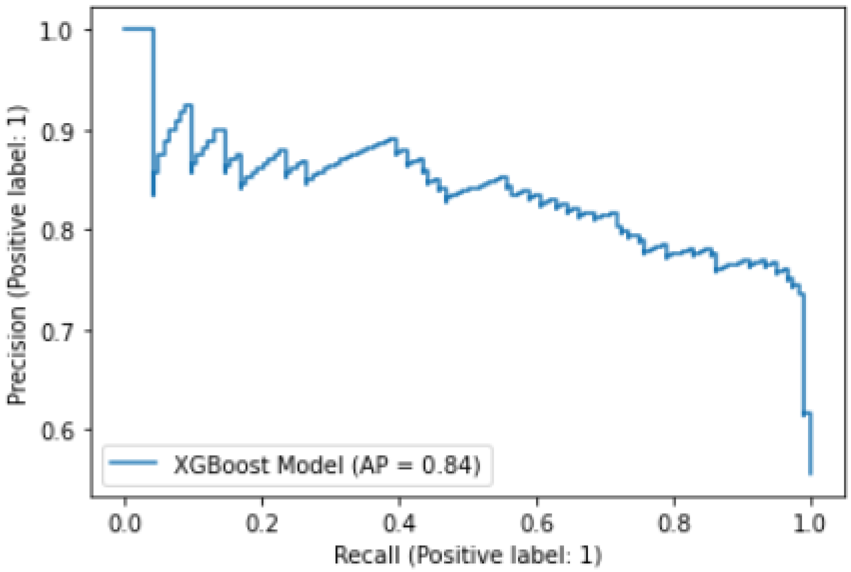
```
In [35]: from sklearn.ensemble import  GradientBoostingClassifier
         from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score,plot_precision_recall_curve,plot_roc_c
         def xgboost(x_train, x_test, y_train, y_test):
             xg = GradientBoostingClassifier()
             xg.fit(x_train,y_train)
             yPred = xg.predict(x_test)
             print("****Gradient BoostingClassifier****")
             print("Confusion matrix")
             confusion_matrix = metrics.confusion_matrix(y_test, yPred)
             cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
             cm_display.plot()
             plt.show()
             print("Classification report")
             print(classification_report (y_test, yPred))
             y_pred=xg.predict(x_test)
             y_pred1=xg.predict(x_train)
             print('Testing accuracy: ',accuracy_score(y_test,y_pred))
             print('Training accuracy: ',accuracy_score(y_train,y_pred1))
             print('AUC Score : ',roc_auc_score(y_test,y_pred))
             plot_precision_recall_curve(xg, x_test, y_test, name = 'XGBoost Model')
             plot_roc_curve(xg, x_test, y_test, name = 'XGBoost Model')

         xgboost(x_train, x_test, y_train, y_test)
```

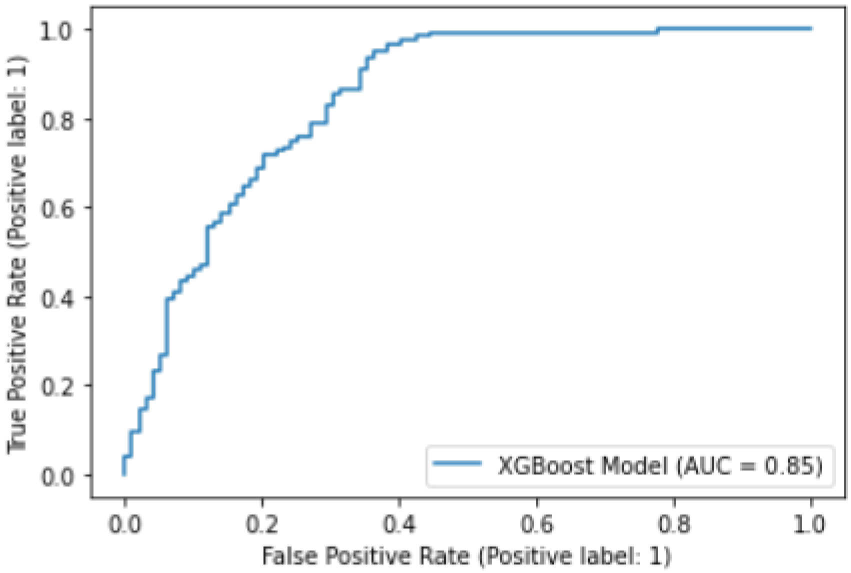
OUTPUT

```
****Gradient BoostingClassifier****
Confusion matrix
```



```
Classification report
              precision    recall  f1-score   support

           0       0.77      0.70      0.73        99
           1       0.77      0.83      0.80       124

    accuracy                           0.77       223
   macro avg       0.77      0.76      0.77       223
weighted avg       0.77      0.77      0.77       223

Testing accuracy:  0.7713004484304933
Training accuracy:  0.9444444444444444
AUC Score :  0.7638074291300098
```


1.0
0.9
0.8
0.7
0.6
Precision (Positive label: 1)
XGBoost Model (AP = 0.84)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
XGBoost Model (AUC = 0.85)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)