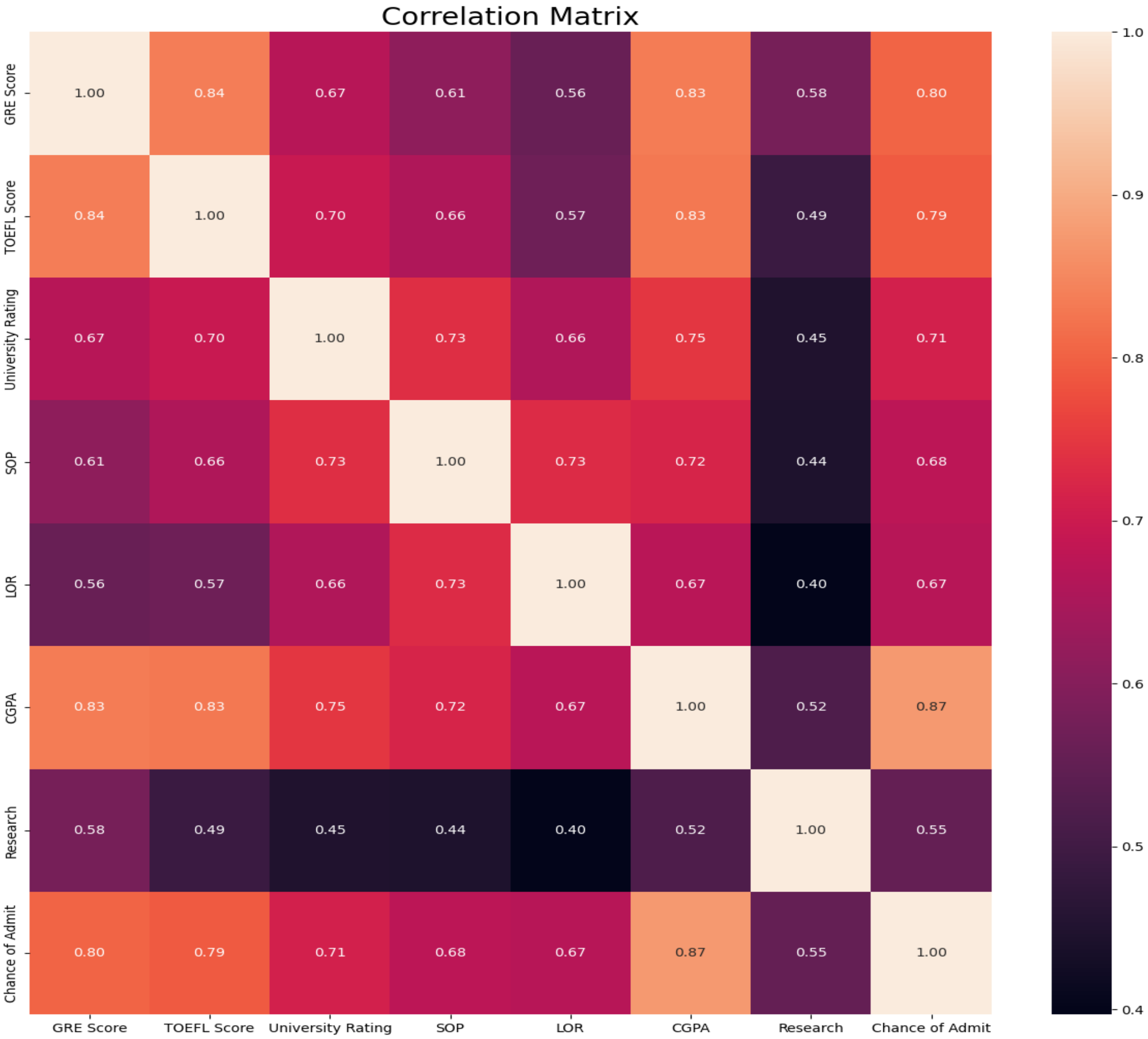
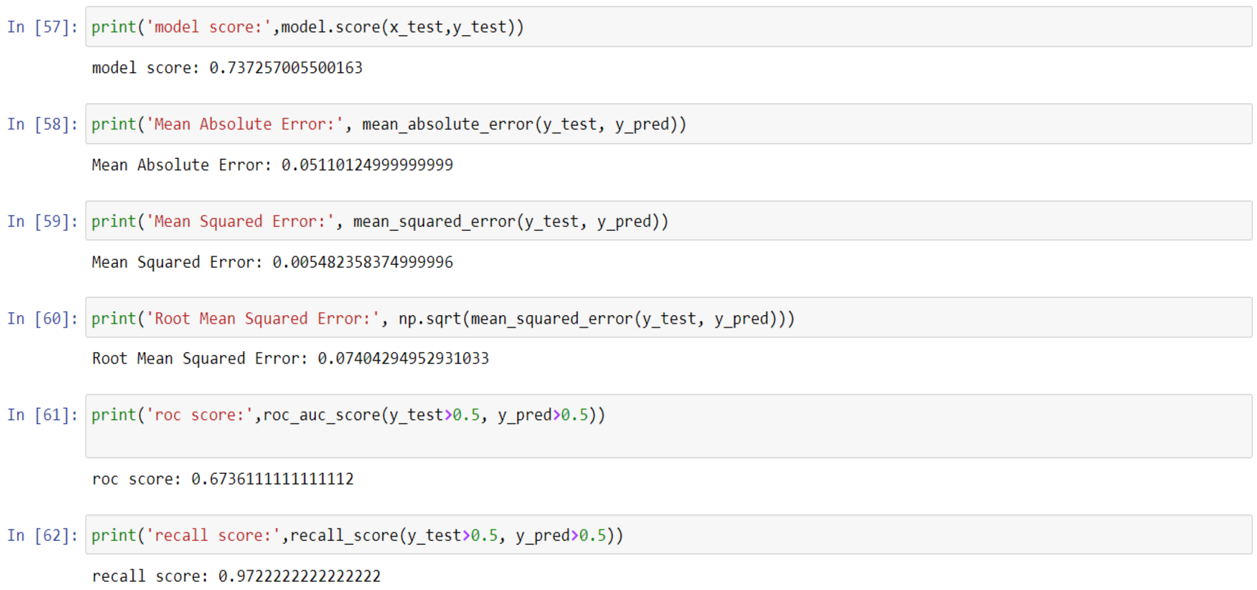
Project Development Phase
Performance Test

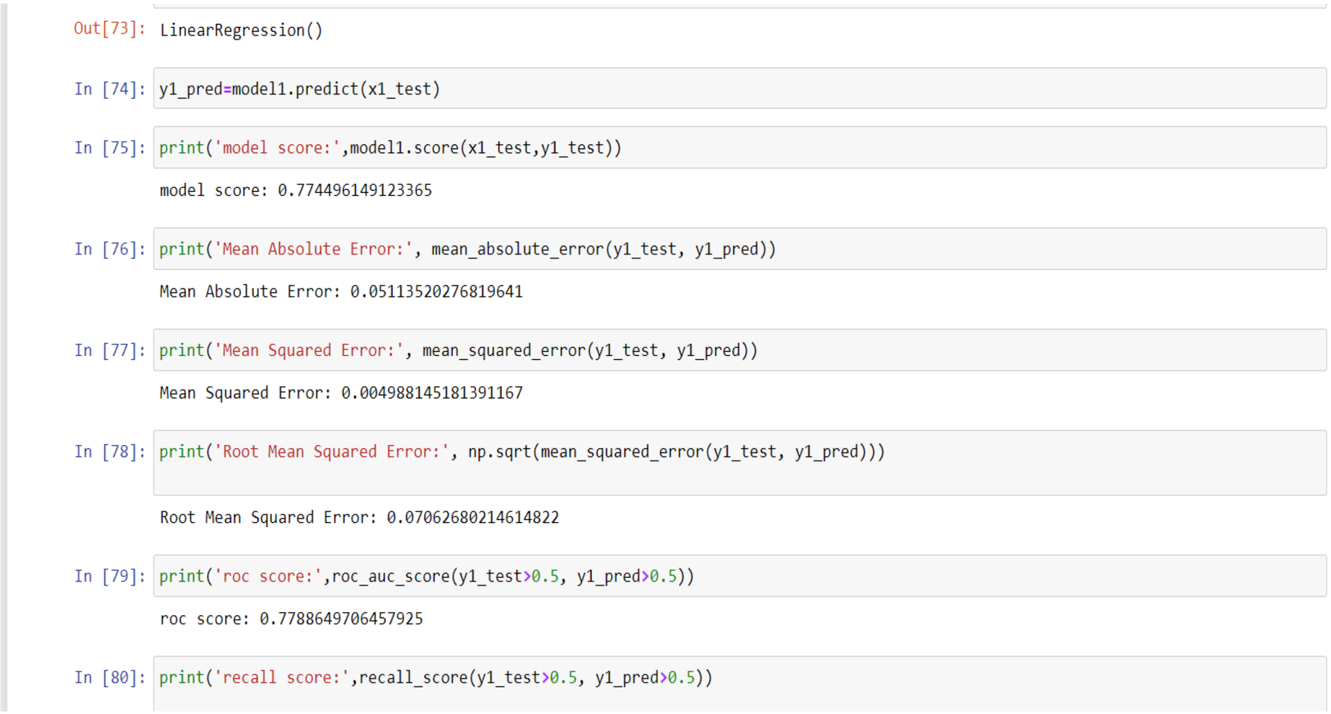
Date	**17 November 2022**
Team ID	**PNT2022TMID37882**
Project Name	**Project – University Admit Eligibility Predictor**
Maximum Marks	**10 Marks**

Some of the Performance testing Screenshots:

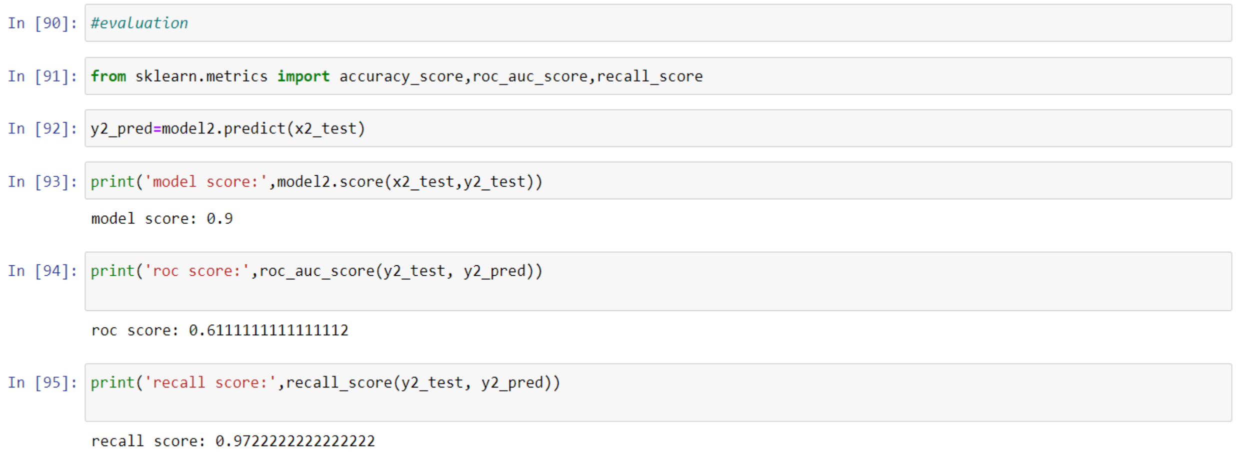




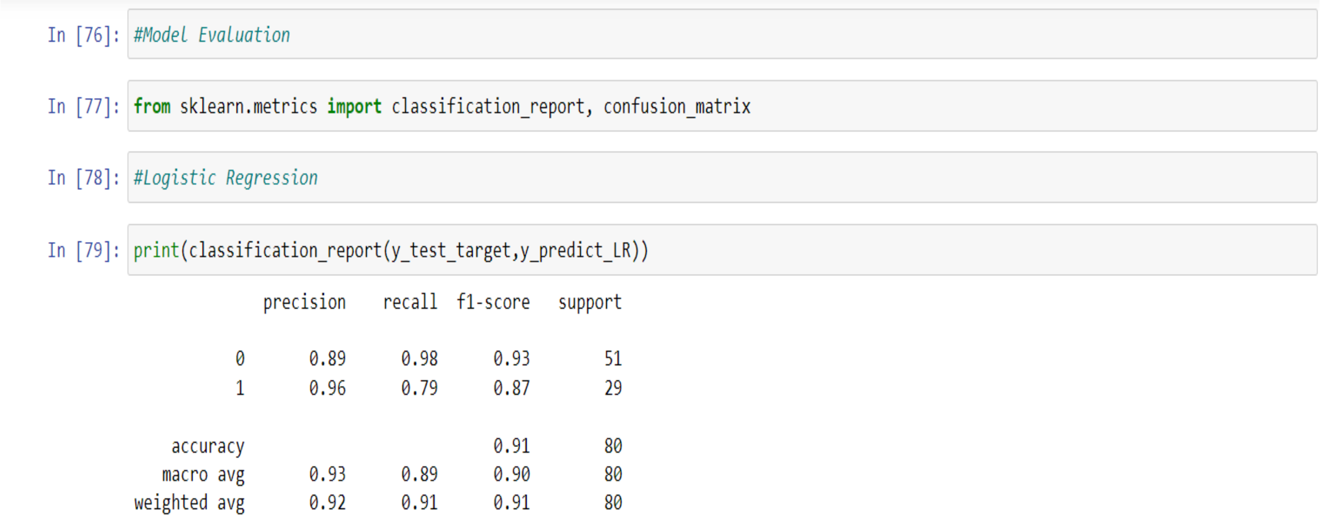
```
In [57]: print('model score:',model.score(x_test,y_test))

         model score: 0.737257005500163

In [58]: print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))

         Mean Absolute Error: 0.05110124999999999

In [59]: print('Mean Squared Error:', mean_squared_error(y_test, y_pred))

         Mean Squared Error: 0.005482358374999996

In [60]: print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))

         Root Mean Squared Error: 0.07404294952931033

In [61]: print('roc score:',roc_auc_score(y_test>0.5, y_pred>0.5))

         roc score: 0.6736111111111112

In [62]: print('recall score:',recall_score(y_test>0.5, y_pred>0.5))

         recall score: 0.9722222222222222
```



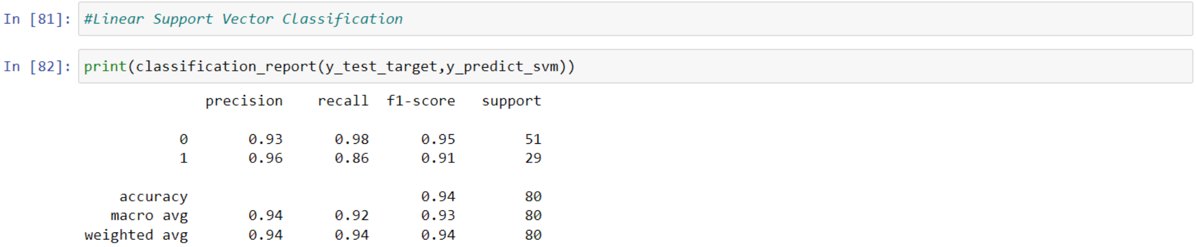
```
Out[73]: LinearRegression()

In [74]: y1_pred=model1.predict(x1_test)

In [75]: print('model score:',model1.score(x1_test,y1_test))

         model score: 0.774496149123365

In [76]: print('Mean Absolute Error:', mean_absolute_error(y1_test, y1_pred))

         Mean Absolute Error: 0.05113520276819641

In [77]: print('Mean Squared Error:', mean_squared_error(y1_test, y1_pred))

         Mean Squared Error: 0.004988145181391167

In [78]: print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y1_test, y1_pred)))

         Root Mean Squared Error: 0.07062680214614822

In [79]: print('roc score:',roc_auc_score(y1_test>0.5, y1_pred>0.5))

         roc score: 0.7788649706457925

In [80]: print('recall score:',recall_score(y1_test>0.5, y1_pred>0.5))
```



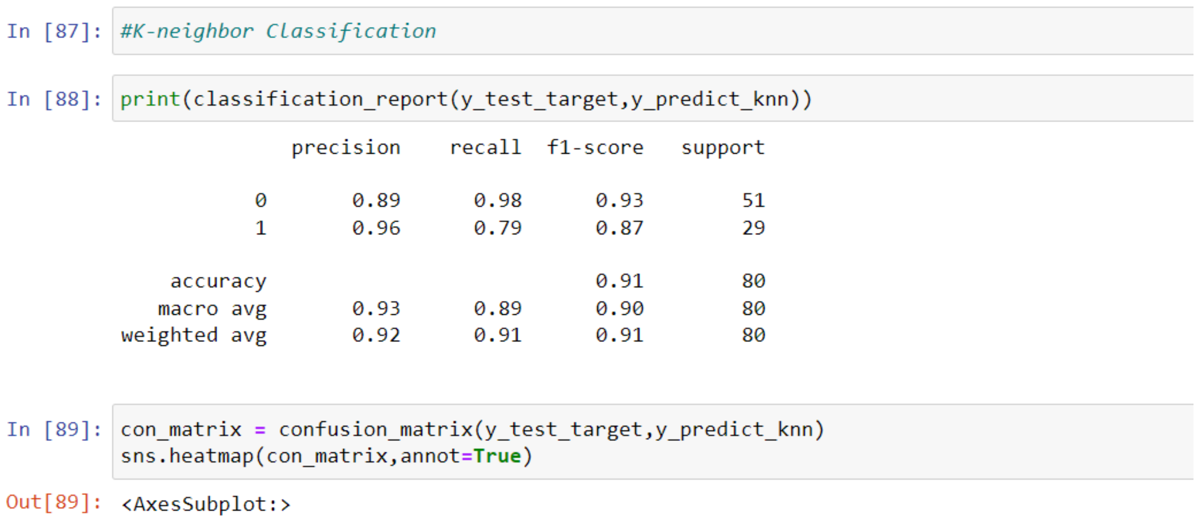
```
In [90]: #evaluation

In [91]: from sklearn.metrics import accuracy_score,roc_auc_score,recall_score

In [92]: y2_pred=model2.predict(x2_test)

In [93]: print('model score:',model2.score(x2_test,y2_test))

         model score: 0.9

In [94]: print('roc score:',roc_auc_score(y2_test, y2_pred))

         roc score: 0.6111111111111112

In [95]: print('recall score:',recall_score(y2_test, y2_pred))

         recall score: 0.9722222222222222
```



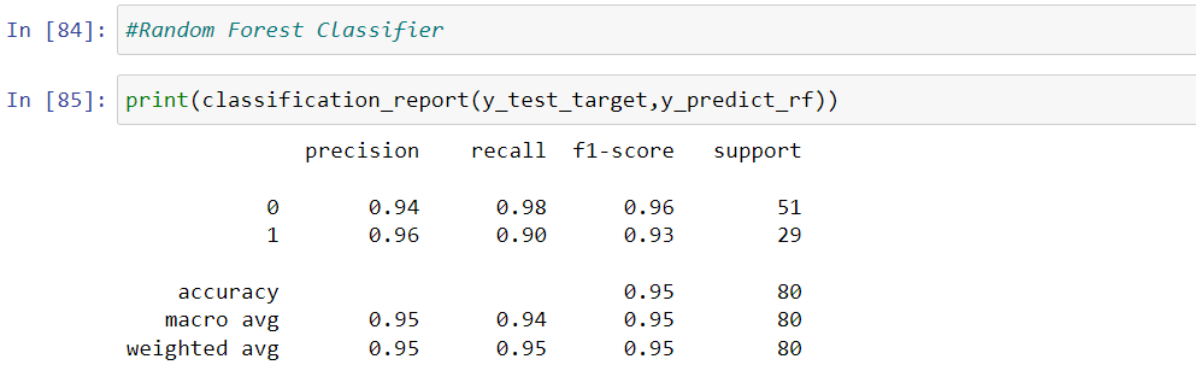
```
In [76]: #Model Evaluation

In [77]: from sklearn.metrics import classification_report, confusion_matrix

In [78]: #Logistic Regression

In [79]: print(classification_report(y_test_target,y_predict_LR))

              precision    recall  f1-score   support

           0       0.89      0.98      0.93        51
           1       0.96      0.79      0.87        29

    accuracy                           0.91        80
   macro avg       0.93      0.89      0.90        80
weighted avg       0.92      0.91      0.91        80
```



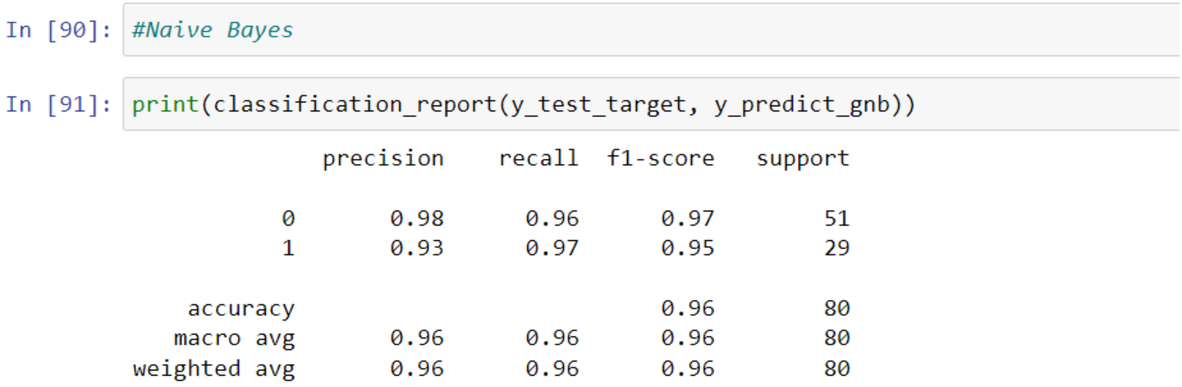
```
In [81]: #Linear Support Vector Classification

In [82]: print(classification_report(y_test_target,y_predict_svm))
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        51
           1       0.96      0.86      0.91        29

    accuracy                           0.94        80
   macro avg       0.94      0.92      0.93        80
weighted avg       0.94      0.94      0.94        80
```



```
In [87]: #K-neighbor Classification

In [88]: print(classification_report(y_test_target,y_predict_knn))

              precision    recall  f1-score   support

           0       0.89      0.98      0.93        51
           1       0.96      0.79      0.87        29

    accuracy                           0.91        80
   macro avg       0.93      0.89      0.90        80
weighted avg       0.92      0.91      0.91        80

In [89]: con_matrix = confusion_matrix(y_test_target,y_predict_knn)
         sns.heatmap(con_matrix,annot=True)

Out[89]: <AxesSubplot:>
```



```
In [84]: #Random Forest Classifier

In [85]: print(classification_report(y_test_target,y_predict_rf))

              precision    recall  f1-score   support

           0       0.94      0.98      0.96        51
           1       0.96      0.90      0.93        29

    accuracy                           0.95        80
   macro avg       0.95      0.94      0.95        80
weighted avg       0.95      0.95      0.95        80
```



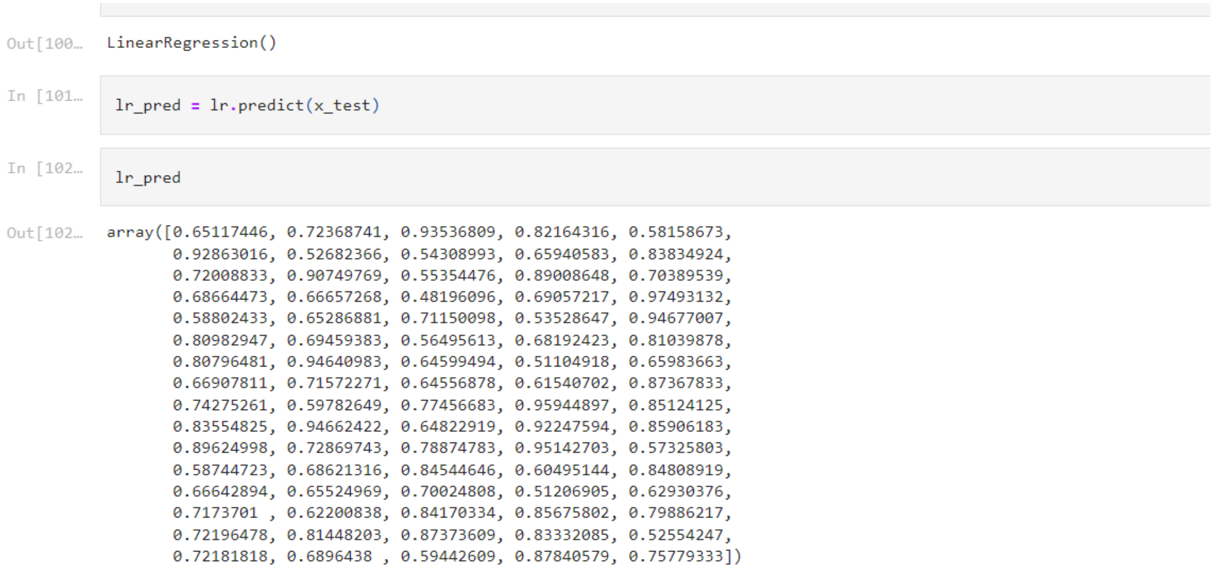
```
In [90]: #Naive Bayes

In [91]: print(classification_report(y_test_target, y_predict_gnb))

              precision    recall  f1-score   support

           0       0.98      0.96      0.97        51
           1       0.93      0.97      0.95        29

    accuracy                           0.96        80
   macro avg       0.96      0.96      0.96        80
weighted avg       0.96      0.96      0.96        80
```

```
In [93]: #Save the Model
```

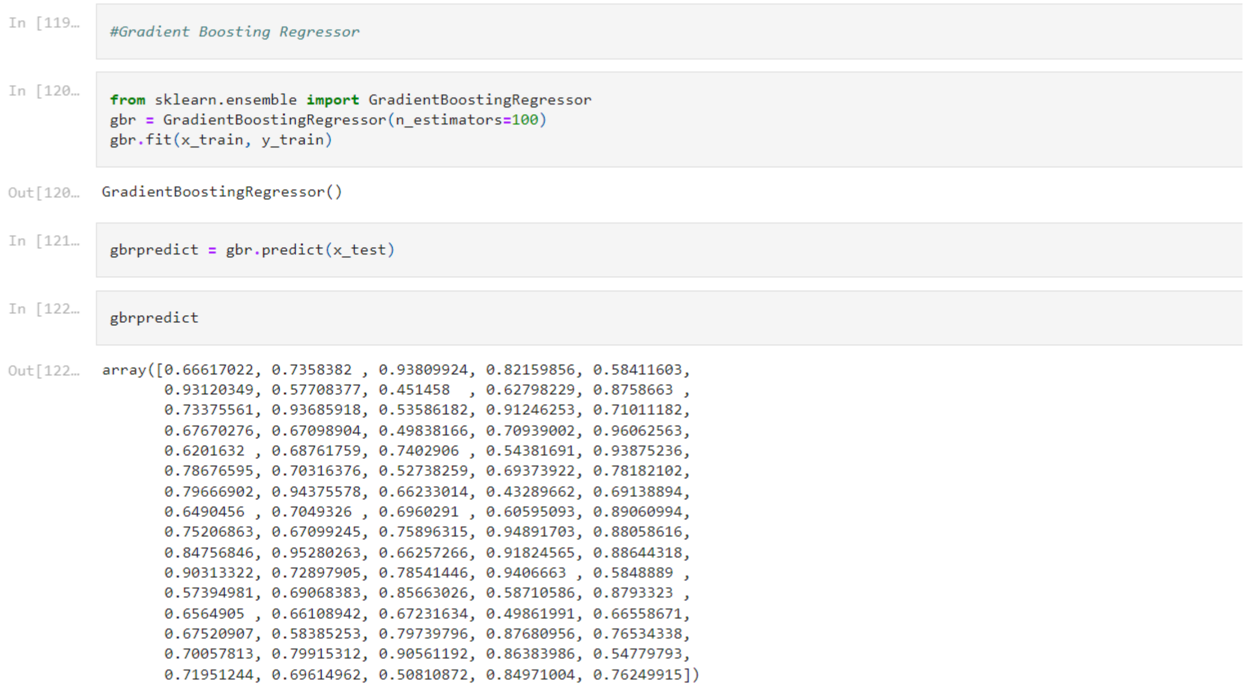
```
In [94]: print(classification_report(y_test_target, y_predict_gnb))
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        51
           1       0.93      0.97      0.95        29

    accuracy                           0.96        80
   macro avg       0.96      0.96      0.96        80
weighted avg       0.96      0.96      0.96        80
```

```
In [95]: #The following scores are the results of the Naive bayes model

         #Accuracy: ~96% label accuracy
         #Precision: ~98% labeled as no chance of admit and ~93% labeled as chance of admit
         #Recall: ~96% labeled as no chance of admit and ~97% labeled as no chance of admit
```



```
Out[100… LinearRegression()

In [101… lr_pred = lr.predict(x_test)

In [102… lr_pred

Out[102… array([0.65117446, 0.72368741, 0.93536809, 0.82164316, 0.58158673,
                0.92863016, 0.52682366, 0.54308993, 0.65940583, 0.83834924,
                0.72008833, 0.90749769, 0.55354476, 0.89008648, 0.70389539,
                0.68664473, 0.66657268, 0.48196096, 0.69057217, 0.97493132,
                0.58802433, 0.65286881, 0.71150098, 0.53528647, 0.94677007,
                0.80982947, 0.69459383, 0.56495613, 0.68192423, 0.81039878,
                0.80796481, 0.94640983, 0.64599494, 0.51104918, 0.65983663,
                0.66907811, 0.71572271, 0.64556878, 0.61540702, 0.87367833,
                0.74275261, 0.59782649, 0.77456683, 0.95944897, 0.85124125,
                0.83554825, 0.94662422, 0.64822919, 0.92247594, 0.85906183,
                0.89624998, 0.72869743, 0.78874783, 0.95142703, 0.57325803,
                0.58744723, 0.68621316, 0.84544646, 0.60495144, 0.84808919,
                0.66642894, 0.65524969, 0.70024808, 0.51206905, 0.62930376,
                0.7173701 , 0.62200838, 0.84170334, 0.85675802, 0.79886217,
                0.72196478, 0.81448203, 0.87373609, 0.83332085, 0.52554247,
                0.72181818, 0.6896438 , 0.59442609, 0.87840579, 0.75779333])
```



```
In [119…
#Gradient Boosting Regressor
```

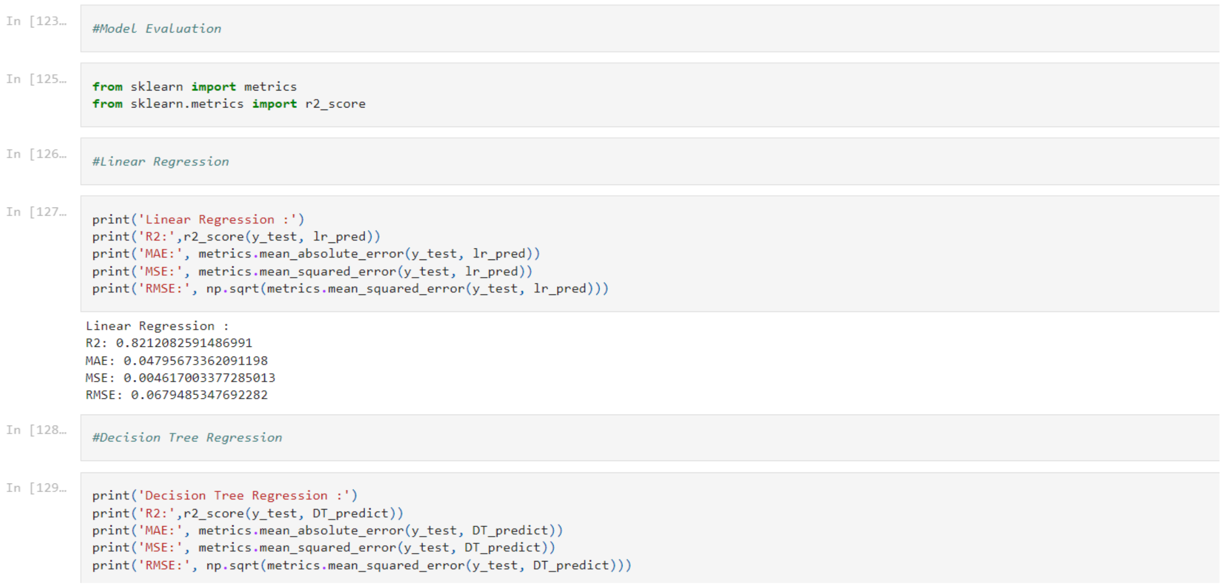
```
In [120…
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor(n_estimators=100)
gbr.fit(x_train, y_train)
```

```
Out[120…
GradientBoostingRegressor()
```

```
In [121…
gbrpredict = gbr.predict(x_test)
```

```
In [122…
gbrpredict
```

```
Out[122…
array([0.66617022, 0.7358382 , 0.93809924, 0.82159856, 0.58411603,
       0.93120349, 0.57708377, 0.451458  , 0.62798229, 0.8758663 ,
       0.73375561, 0.93685918, 0.53586182, 0.91246253, 0.71011182,
       0.67670276, 0.67098904, 0.49838166, 0.70939002, 0.96062563,
       0.6201632 , 0.68761759, 0.7402906 , 0.54381691, 0.93875236,
       0.78676595, 0.70316376, 0.52738259, 0.69373922, 0.78182102,
       0.79666902, 0.94375578, 0.66233014, 0.43289662, 0.69138894,
       0.6490456 , 0.7049326 , 0.6960291 , 0.60595093, 0.89060994,
       0.75206863, 0.67099245, 0.75896315, 0.94891703, 0.88058616,
       0.84756846, 0.95280263, 0.66257266, 0.91824565, 0.88644318,
       0.90313322, 0.72897905, 0.78541446, 0.9406663 , 0.5848889 ,
       0.57394981, 0.69068383, 0.85663026, 0.58710586, 0.8793323 ,
       0.6564905 , 0.66108942, 0.67231634, 0.49861991, 0.66558671,
       0.67520907, 0.58385253, 0.79739796, 0.87680956, 0.76534338,
       0.70057813, 0.79915312, 0.90561192, 0.86383986, 0.54779793,
       0.71951244, 0.69614962, 0.50810872, 0.84971004, 0.76249915])
```



```
In [123…
#Model Evaluation
```

```
In [125…
from sklearn import metrics
from sklearn.metrics import r2_score
```

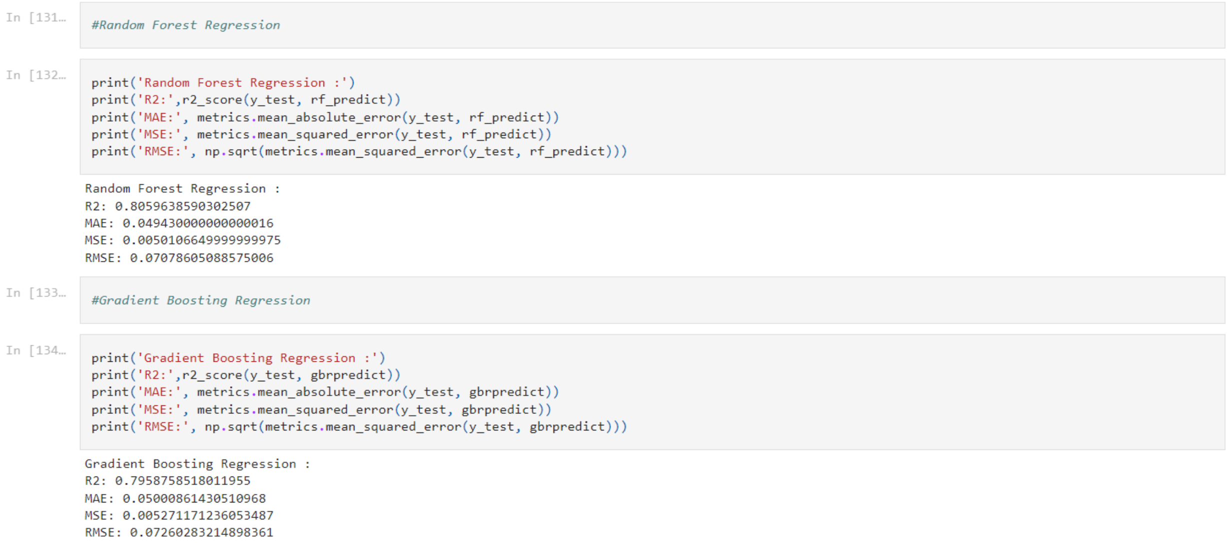
```
In [126…
#Linear Regression
```

```
In [127…
print('Linear Regression :')
print('R2:',r2_score(y_test, lr_pred))
print('MAE:', metrics.mean_absolute_error(y_test, lr_pred))
print('MSE:', metrics.mean_squared_error(y_test, lr_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, lr_pred)))
```

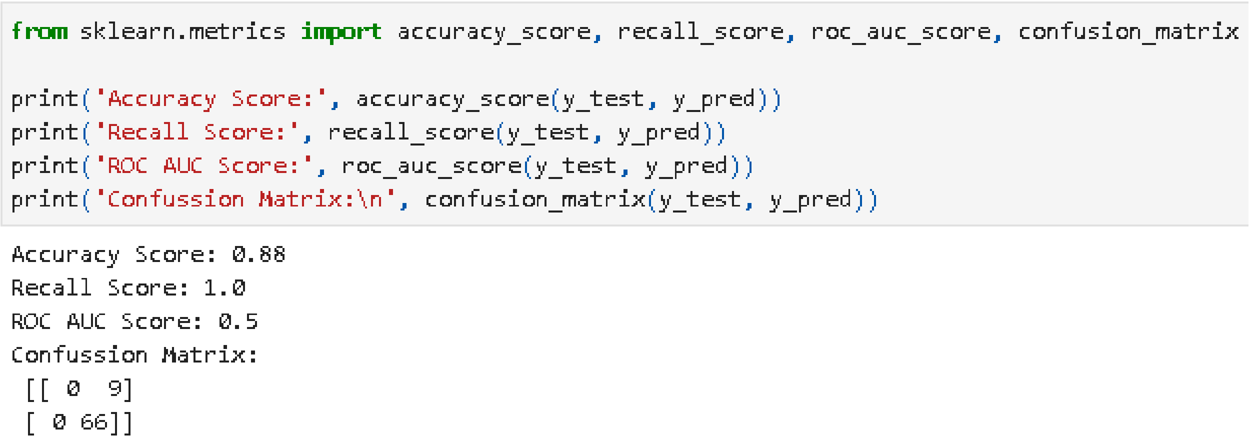
```
Linear Regression :
R2: 0.8212082591486991
MAE: 0.04795673362091198
MSE: 0.004617003377285013
RMSE: 0.0679485347692282
```

```
In [128…
#Decision Tree Regression
```

```
In [129…
print('Decision Tree Regression :')
print('R2:',r2_score(y_test, DT_predict))
print('MAE:', metrics.mean_absolute_error(y_test, DT_predict))
print('MSE:', metrics.mean_squared_error(y_test, DT_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, DT_predict)))
```



```
In [131…  #Random Forest Regression

In [132…  print('Random Forest Regression :')
          print('R2:',r2_score(y_test, rf_predict))
          print('MAE:', metrics.mean_absolute_error(y_test, rf_predict))
          print('MSE:', metrics.mean_squared_error(y_test, rf_predict))
          print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, rf_predict)))

          Random Forest Regression :
          R2: 0.8059638590302507
          MAE: 0.049430000000000016
          MSE: 0.0050106649999999975
          RMSE: 0.07078605088575006

In [133…  #Gradient Boosting Regression

In [134…  print('Gradient Boosting Regression :')
          print('R2:',r2_score(y_test, gbrpredict))
          print('MAE:', metrics.mean_absolute_error(y_test, gbrpredict))
          print('MSE:', metrics.mean_squared_error(y_test, gbrpredict))
          print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, gbrpredict)))

          Gradient Boosting Regression :
          R2: 0.7958758518011955
          MAE: 0.05000861430510968
          MSE: 0.005271171236053487
          RMSE: 0.07260283214898361
```



```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix

print('Accuracy Score:', accuracy_score(y_test, y_pred))
print('Recall Score:', recall_score(y_test, y_pred))
print('ROC AUC Score:', roc_auc_score(y_test, y_pred))
print('Confussion Matrix:\n', confusion_matrix(y_test, y_pred))
```

```
Accuracy Score: 0.88
Recall Score: 1.0
ROC AUC Score: 0.5
Confussion Matrix:
 [[ 0  9]
 [ 0 66]]
```