Twitter Sentiment Prediction of US Airlines
Milestone: Performance Evaluation and Interpretation

Group 12

Student-1 Name: Amitoj Singh Kohli

Student-2 Name: Aryaman Kakad

857-313-0650 (Tel of Student 1)

857-313-5234 (Tel of Student 2)

kohli.am@northeastern.edu

kakad.a@northeastern.edu

Percentage of Effort Contributed by Student1: 100%

Percentage of Effort Contributed by Student2: 100%

Signature of Student 1: Amitoj

Signature of Student 2: ASKakad

Submission Date: 04 / 16 / 23

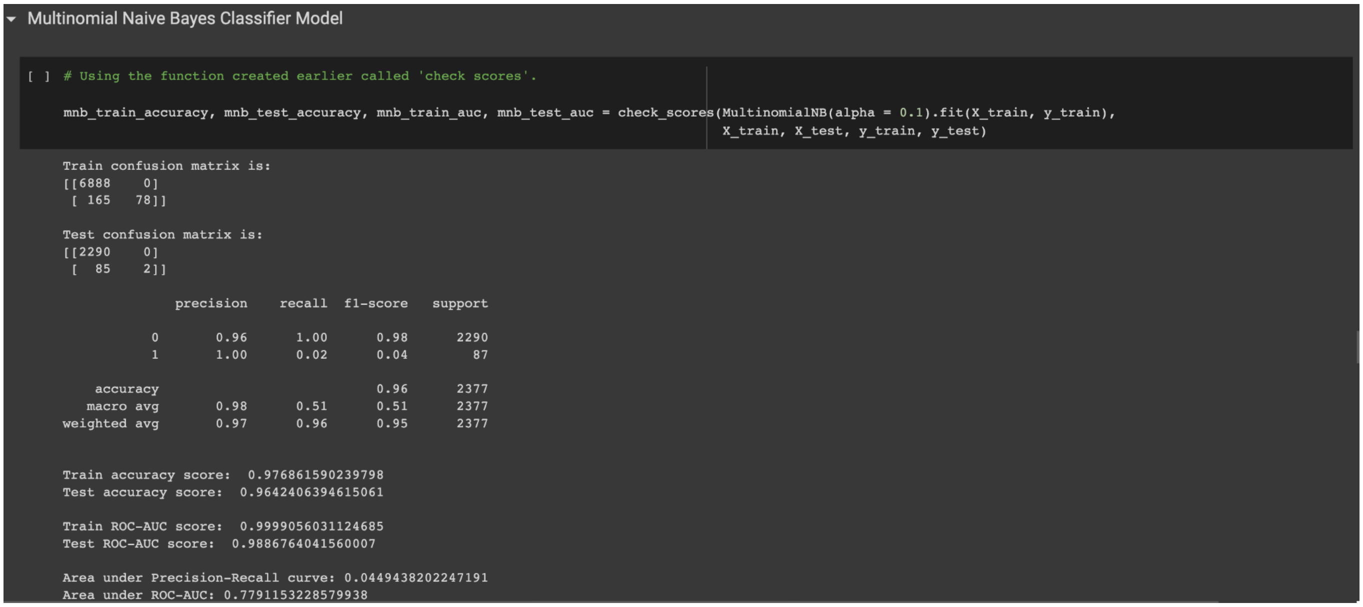
As our problem is related to Classification Supervised Machine Learning. We have applied seven techniques related to classification supervised learning.

1. Naive Bayes Classification
2. Random Forest Classification
3. Logistic Regression
4. K Nearest Neighbors Classification
5. Support Vector Machines
6. Linear Discriminant Analysis
7. Neural Network

We have presented the training and validation accuracy of each of these models as well as the classification summary which includes accuracy, precision, recall and f-1 score.

We have also plotted the Precision vs Recall graph and ROC-AUC Curve for each of the models to decide which out these 7 model is the best for our classification problem.

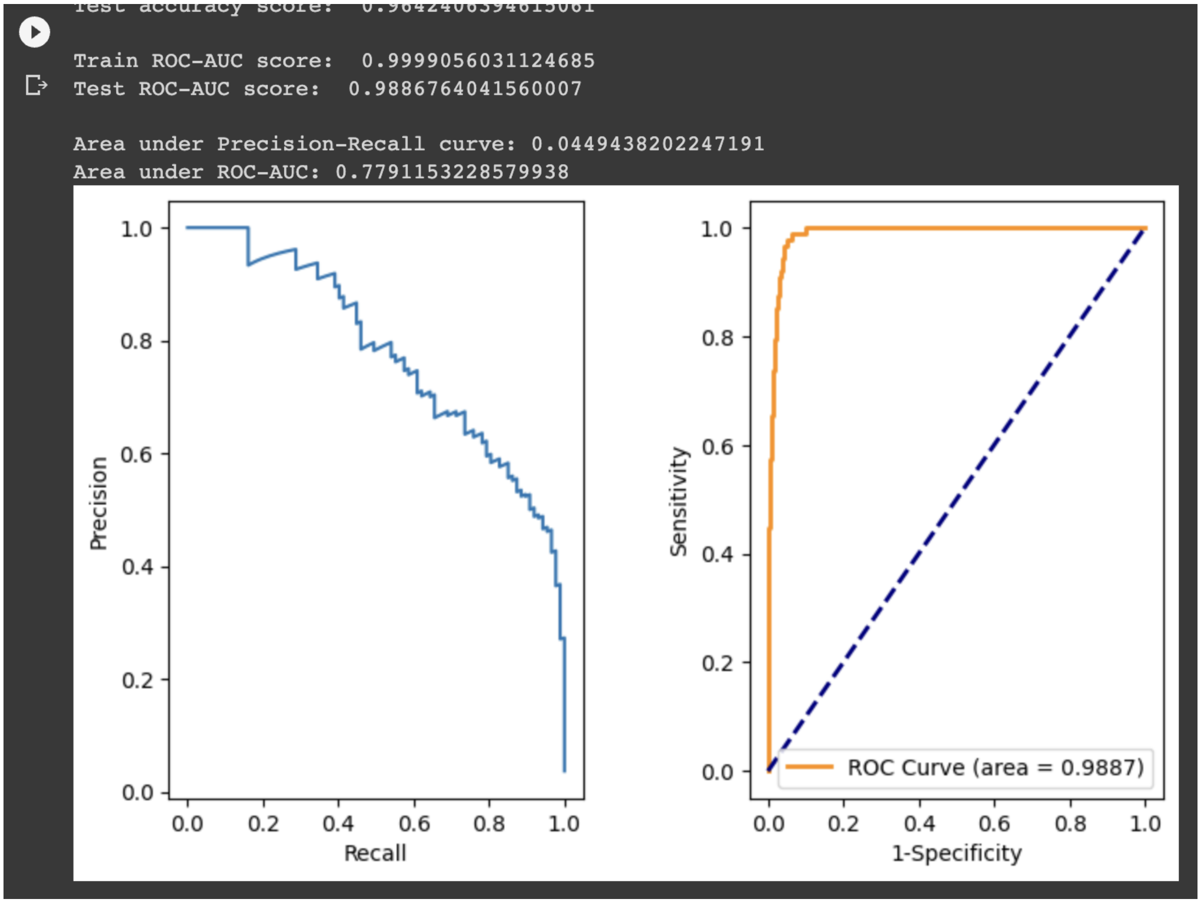
1. **Naive-Bayes Classification**



Multinomial Naive Bayes Classifier Model

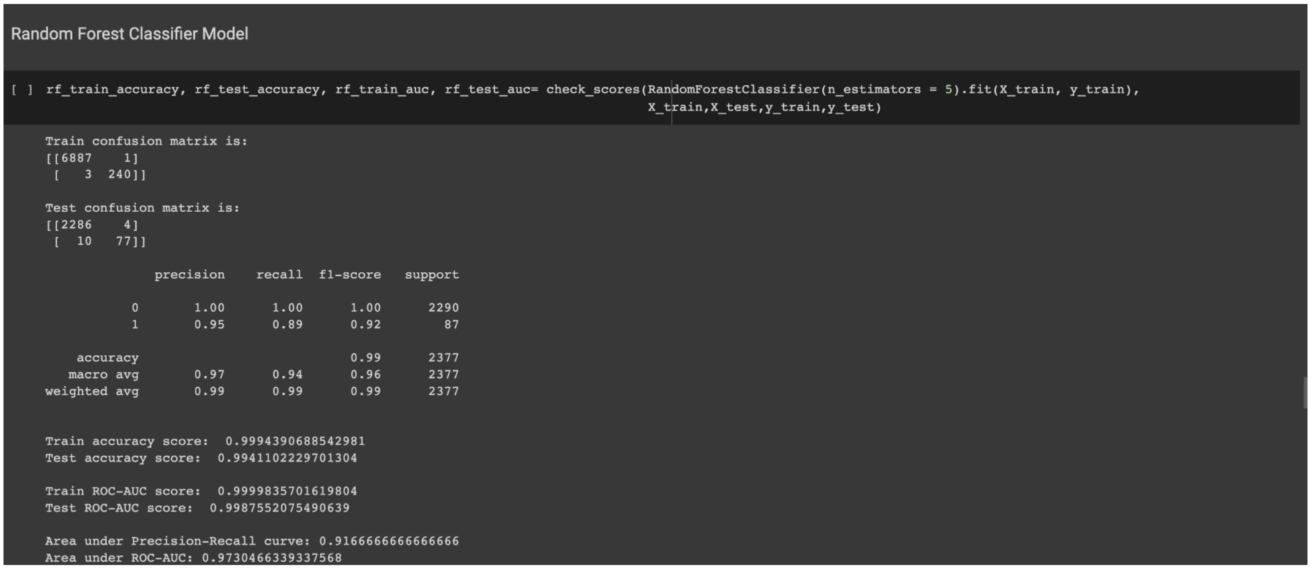
```
[ ] # Using the function created earlier called 'check scores'.

    mnb_train_accuracy, mnb_test_accuracy, mnb_train_auc, mnb_test_auc = check_scores(MultinomialNB(alpha = 0.1).fit(X_train, y_train),
                                                                                      X_train, X_test, y_train, y_test)
```

```
Train confusion matrix is:
[[6888    0]
 [ 165   78]]

Test confusion matrix is:
[[2290    0]
 [  85    2]]

              precision    recall  f1-score   support

           0       0.96      1.00      0.98      2290
           1       1.00      0.02      0.04        87

    accuracy                           0.96      2377
   macro avg       0.98      0.51      0.51      2377
weighted avg       0.97      0.96      0.95      2377


Train accuracy score:  0.976861590239798
Test accuracy score:  0.9642406394615061

Train ROC-AUC score:  0.9999056031124685
Test ROC-AUC score:  0.9886764041560007

Area under Precision-Recall curve: 0.0449438202247191
Area under ROC-AUC: 0.7791153228579938
```



```
Test accuracy score:  0.9642406394615061

Train ROC-AUC score:  0.9999056031124685
Test ROC-AUC score:  0.9886764041560007

Area under Precision-Recall curve: 0.0449438202247191
Area under ROC-AUC: 0.7791153228579938
```

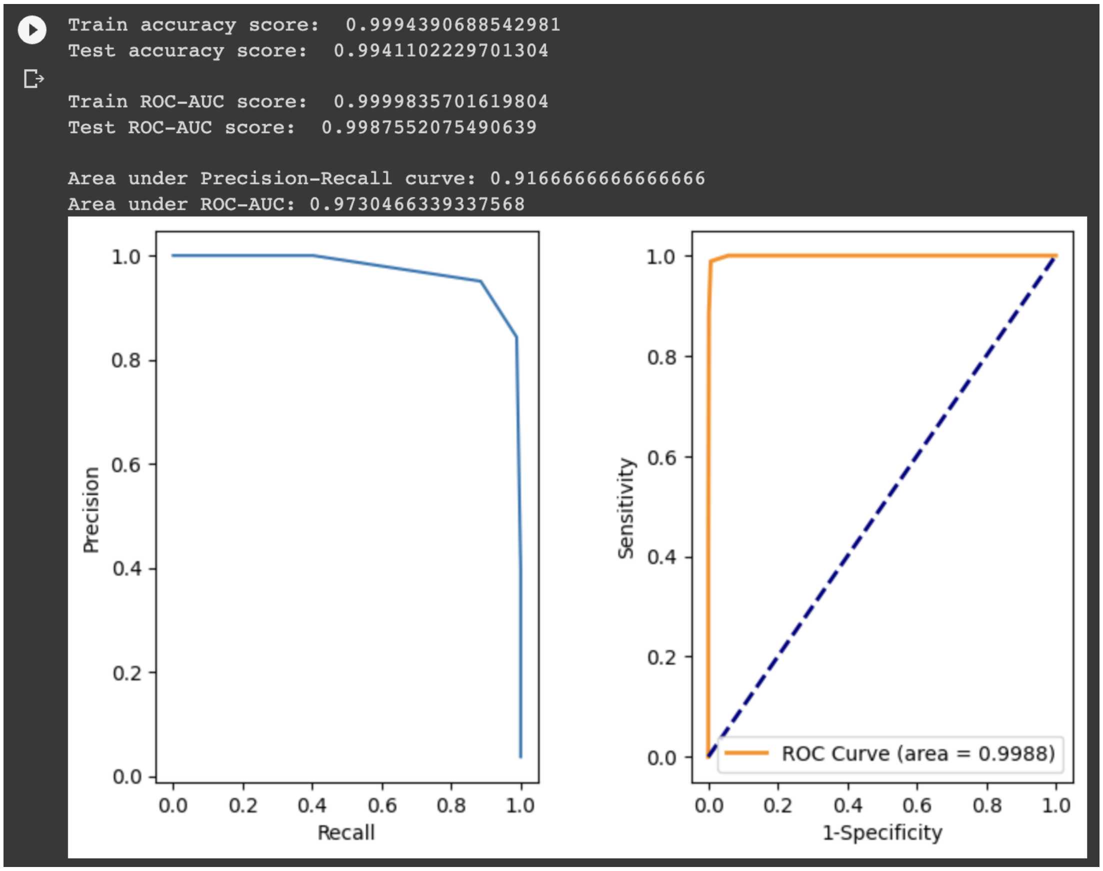
2. Random Forest Classification



Random Forest Classifier Model

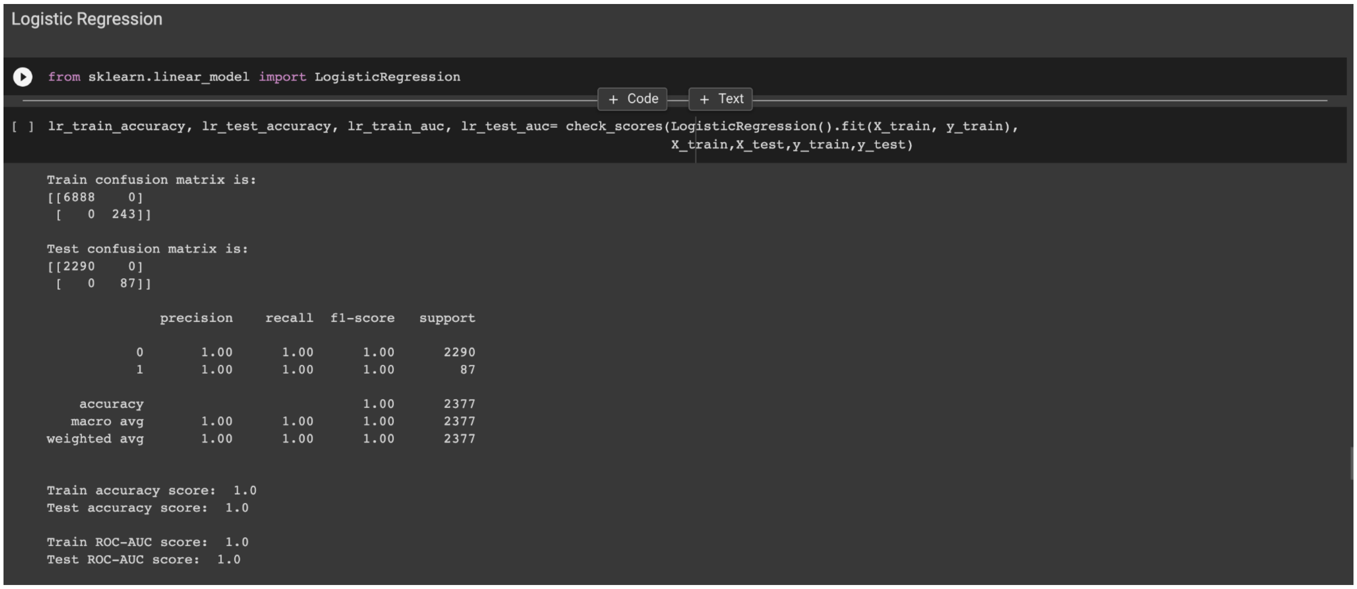
```
[ ]  rf_train_accuracy, rf_test_accuracy, rf_train_auc, rf_test_auc= check_scores(RandomForestClassifier(n_estimators = 5).fit(X_train, y_train),
                                                                                  X_train,X_test,y_train,y_test)
```

```
Train confusion matrix is:
[[6887    1]
 [   3  240]]

Test confusion matrix is:
[[2286    4]
 [  10   77]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2290
           1       0.95      0.89      0.92        87

    accuracy                           0.99      2377
   macro avg       0.97      0.94      0.96      2377
weighted avg       0.99      0.99      0.99      2377


Train accuracy score:  0.9994390688542981
Test accuracy score:  0.9941102229701304

Train ROC-AUC score:  0.9999835701619804
Test ROC-AUC score:  0.9987552075490639

Area under Precision-Recall curve: 0.9166666666666666
Area under ROC-AUC: 0.9730466339337568
```



```
Train accuracy score:  0.9994390688542981
Test accuracy score:  0.9941102229701304

Train ROC-AUC score:  0.9999835701619804
Test ROC-AUC score:  0.9987552075490639

Area under Precision-Recall curve: 0.9166666666666666
Area under ROC-AUC: 0.9730466339337568
```

3. Logistic Regression

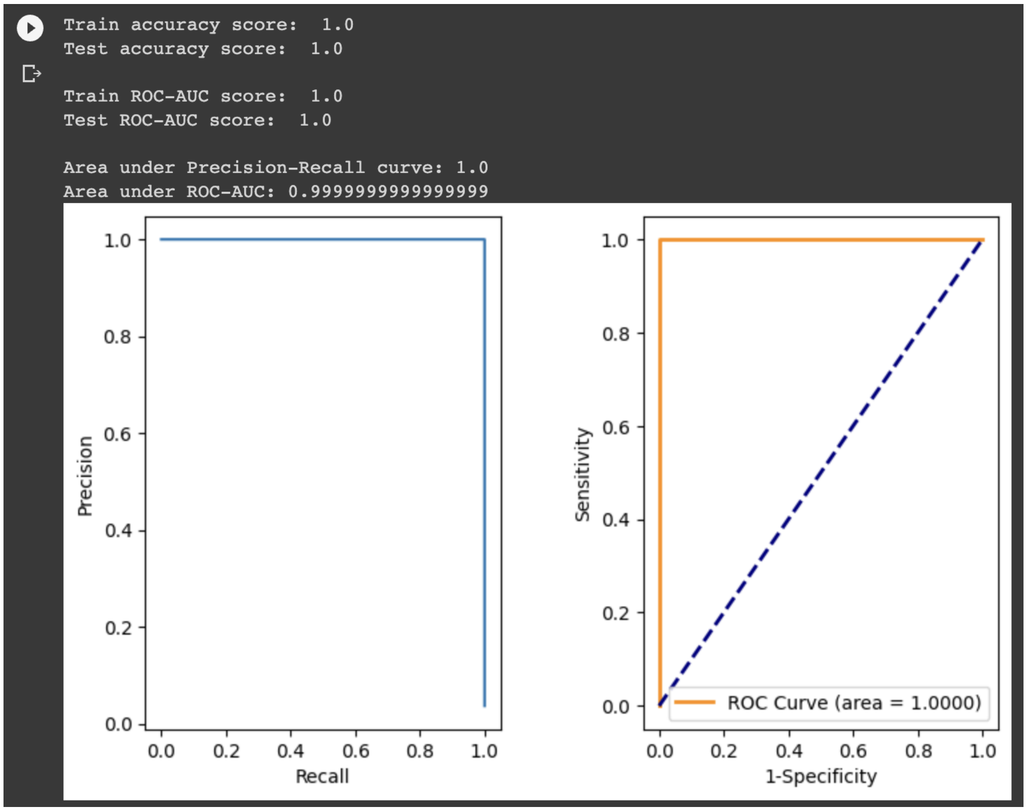


Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

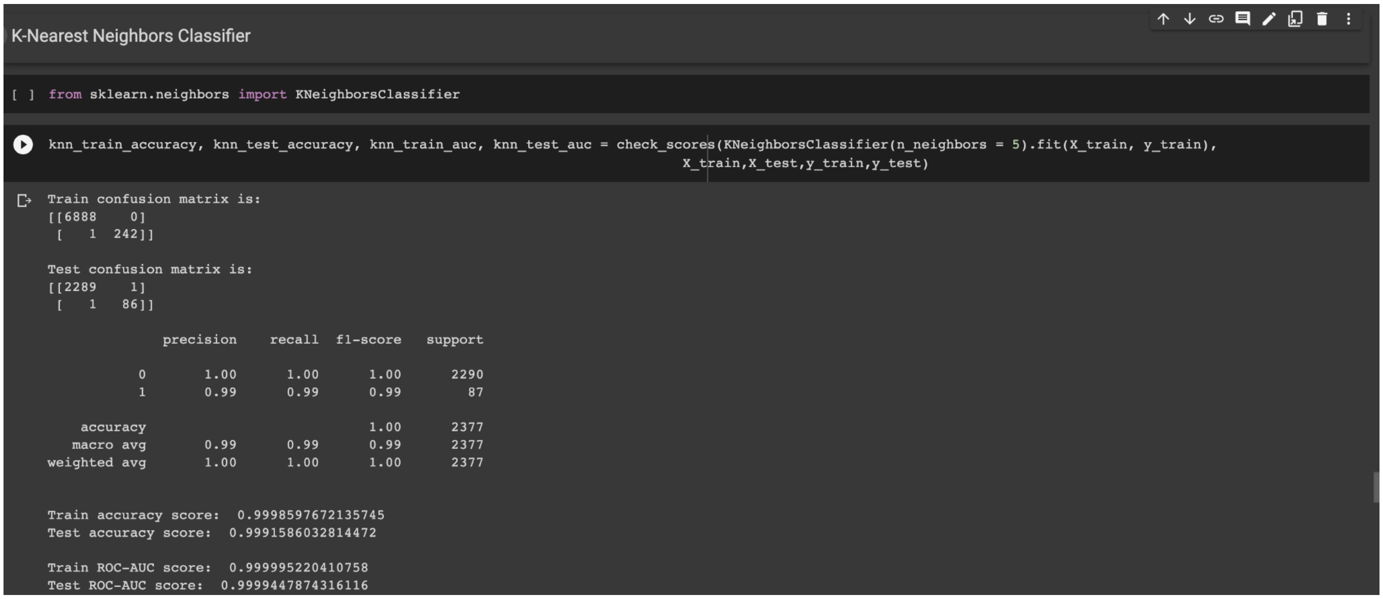
```
[ ] lr_train_accuracy, lr_test_accuracy, lr_train_auc, lr_test_auc= check_scores(LogisticRegression().fit(X_train, y_train),
                                                                                  X_train,X_test,y_train,y_test)
```

```
Train confusion matrix is:
[[6888    0]
 [   0  243]]

Test confusion matrix is:
[[2290    0]
 [   0   87]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2290
           1       1.00      1.00      1.00        87

    accuracy                           1.00      2377
   macro avg       1.00      1.00      1.00      2377
weighted avg       1.00      1.00      1.00      2377


Train accuracy score:  1.0
Test accuracy score:  1.0

Train ROC-AUC score:  1.0
Test ROC-AUC score:  1.0
```



```
Train accuracy score:  1.0
Test accuracy score:  1.0

Train ROC-AUC score:  1.0
Test ROC-AUC score:  1.0

Area under Precision-Recall curve: 1.0
Area under ROC-AUC: 0.9999999999999999
```

4. K-Nearest Neighbors Classification

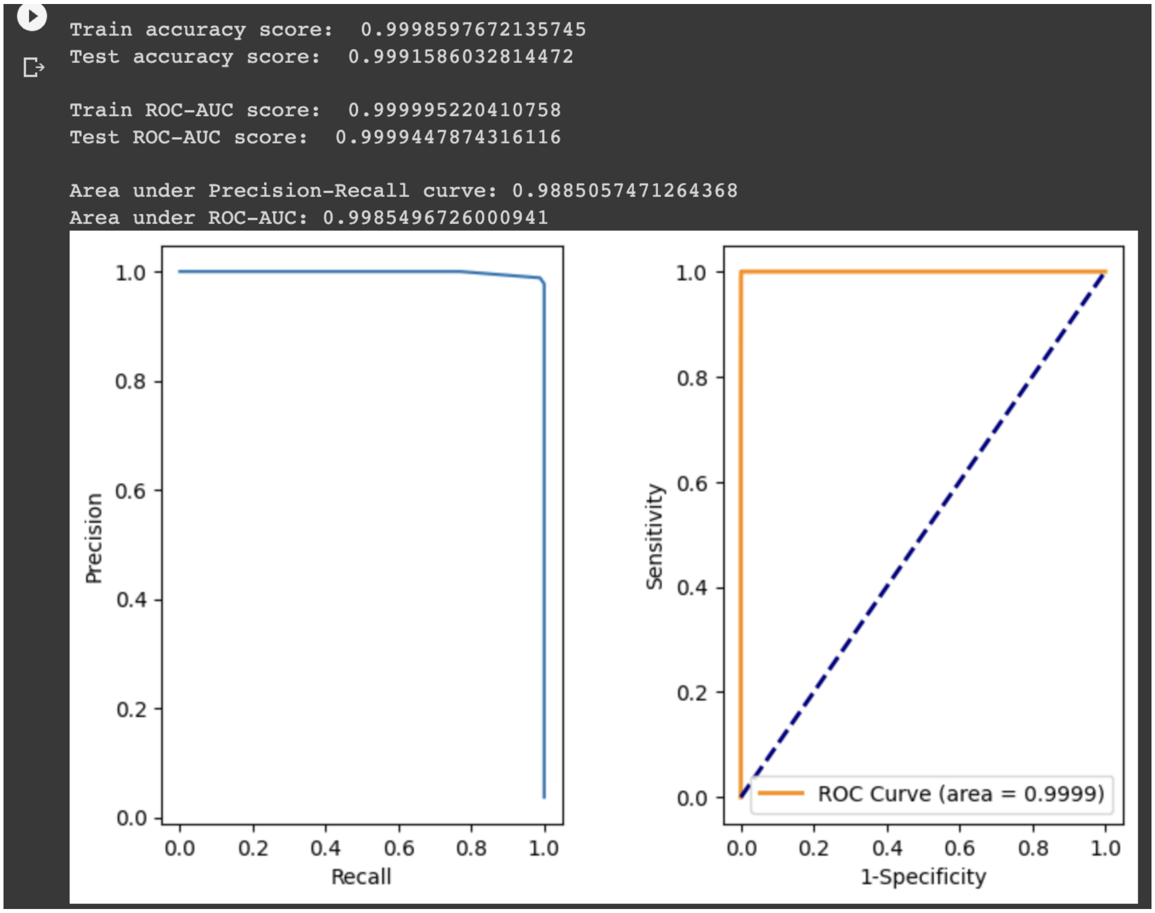


K-Nearest Neighbors Classifier

```
[ ] from sklearn.neighbors import KNeighborsClassifier
```

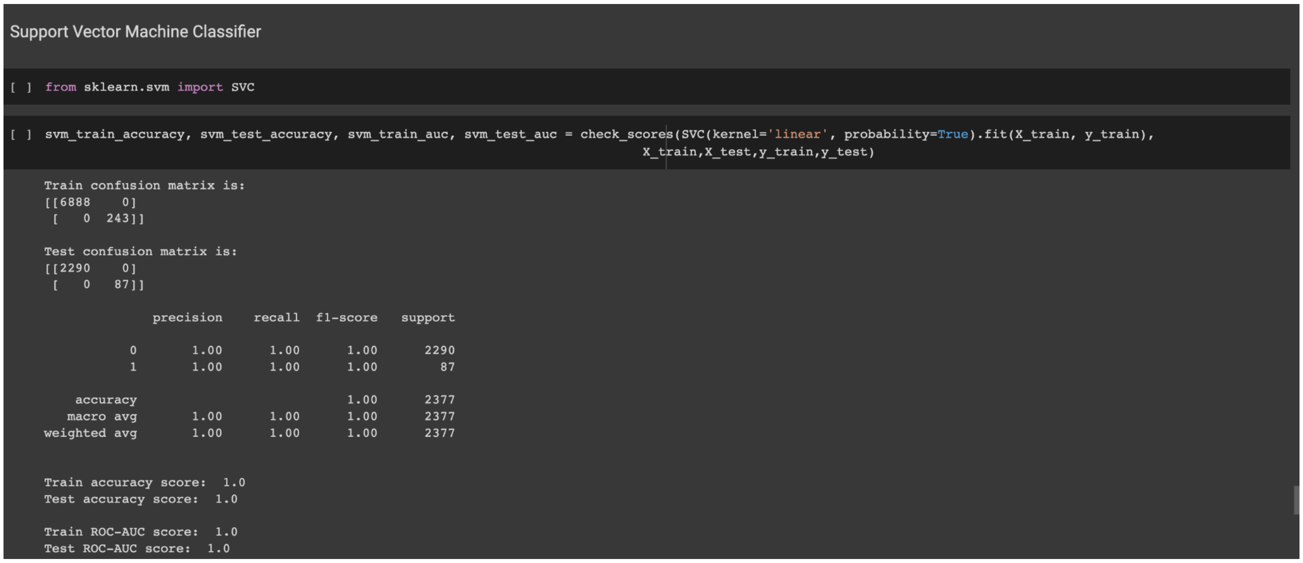
```
knn_train_accuracy, knn_test_accuracy, knn_train_auc, knn_test_auc = check_scores(KNeighborsClassifier(n_neighbors = 5).fit(X_train, y_train),
                                                                              X_train,X_test,y_train,y_test)
```

```
Train confusion matrix is:
[[6888    0]
 [   1  242]]

Test confusion matrix is:
[[2289    1]
 [   1   86]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2290
           1       0.99      0.99      0.99        87

    accuracy                           1.00      2377
   macro avg       0.99      0.99      0.99      2377
weighted avg       1.00      1.00      1.00      2377


Train accuracy score:  0.9998597672135745
Test accuracy score:  0.9991586032814472

Train ROC-AUC score:  0.999995220410758
Test ROC-AUC score:  0.9999447874316116
```



```
Train accuracy score:  0.9998597672135745
Test accuracy score:  0.9991586032814472

Train ROC-AUC score:  0.999995220410758
Test ROC-AUC score:  0.9999447874316116

Area under Precision-Recall curve: 0.9885057471264368
Area under ROC-AUC: 0.9985496726000941
```

5. Support Vector Machines

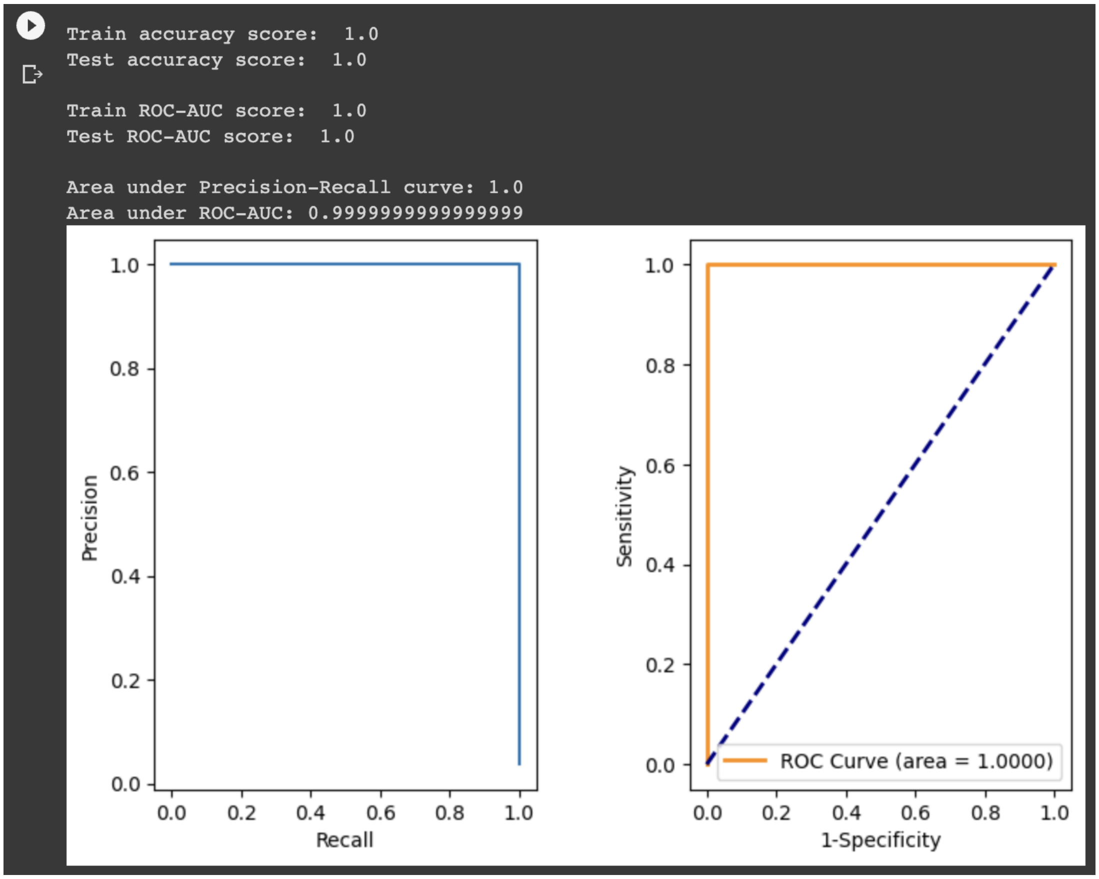


Support Vector Machine Classifier

```
[ ] from sklearn.svm import SVC
```

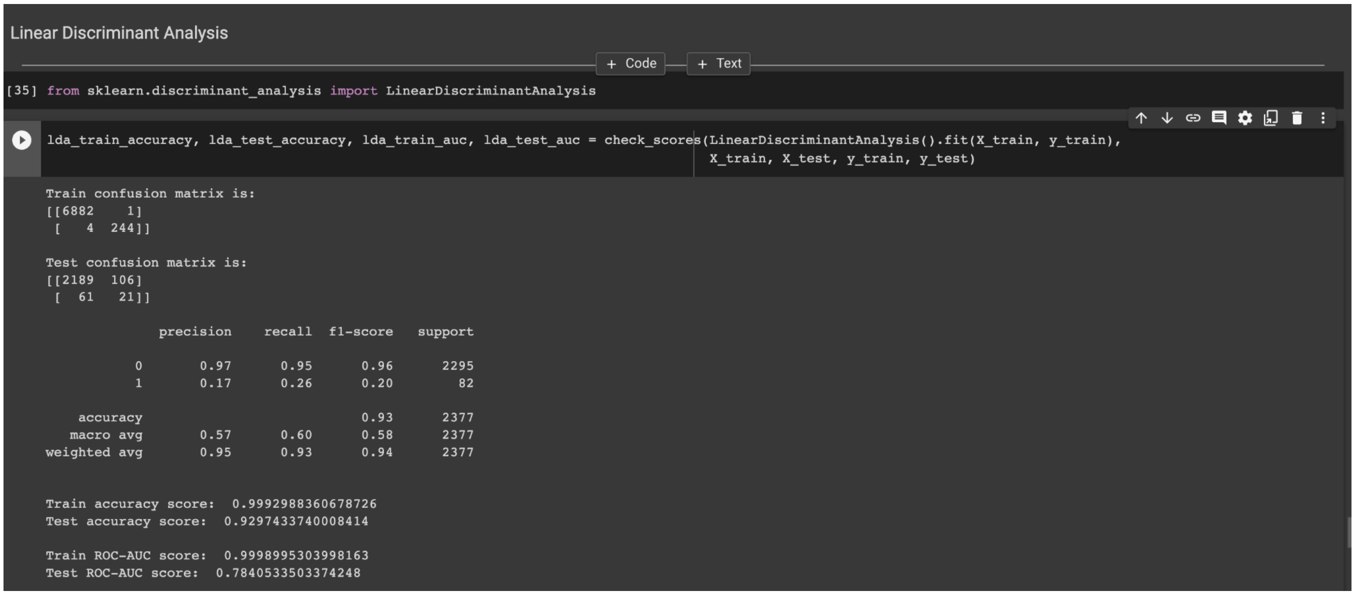
```
[ ] svm_train_accuracy, svm_test_accuracy, svm_train_auc, svm_test_auc = check_scores(SVC(kernel='linear', probability=True).fit(X_train, y_train),
                                                                                      X_train,X_test,y_train,y_test)
```

```
Train confusion matrix is:
[[6888    0]
 [   0  243]]

Test confusion matrix is:
[[2290    0]
 [   0   87]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2290
           1       1.00      1.00      1.00        87

    accuracy                           1.00      2377
   macro avg       1.00      1.00      1.00      2377
weighted avg       1.00      1.00      1.00      2377


Train accuracy score:  1.0
Test accuracy score:  1.0

Train ROC-AUC score:  1.0
Test ROC-AUC score:  1.0
```



```
Train accuracy score:  1.0
Test accuracy score:  1.0

Train ROC-AUC score:  1.0
Test ROC-AUC score:  1.0

Area under Precision-Recall curve: 1.0
Area under ROC-AUC: 0.9999999999999999
```

6. Linear Discriminant Analysis

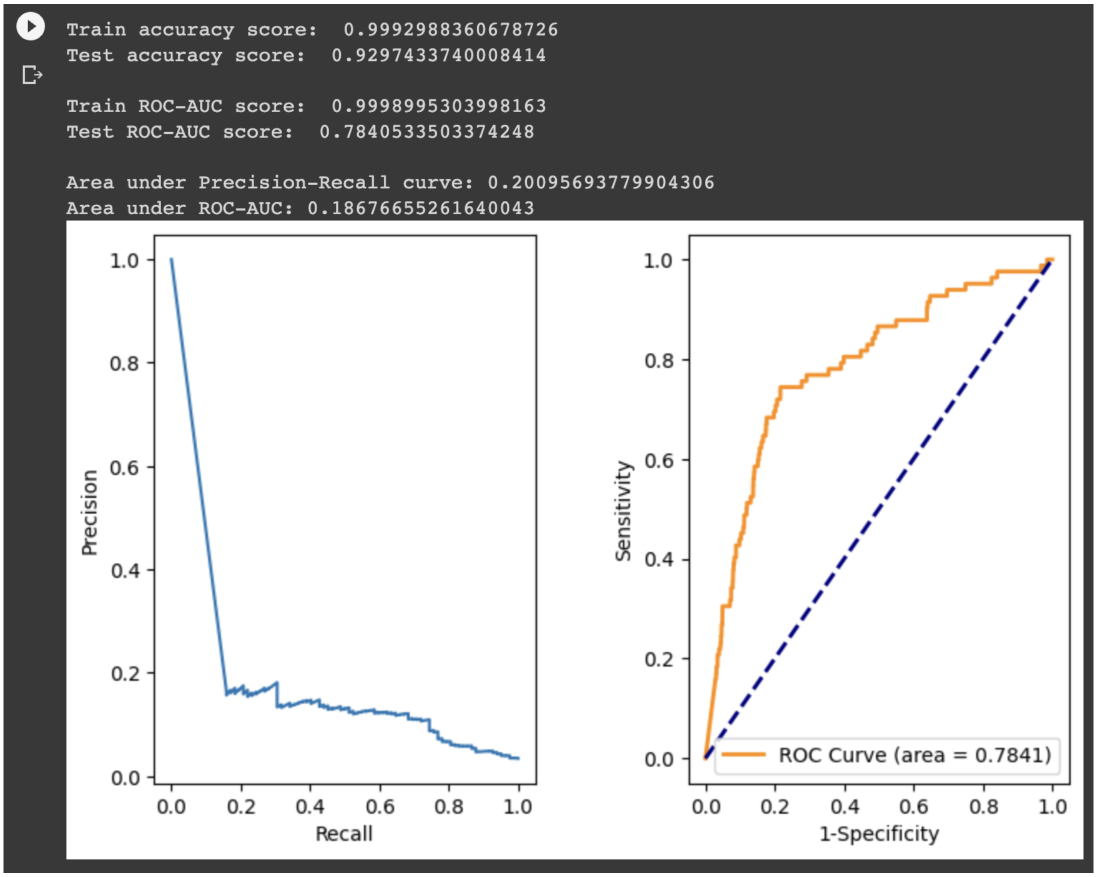


Linear Discriminant Analysis

```
[35] from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

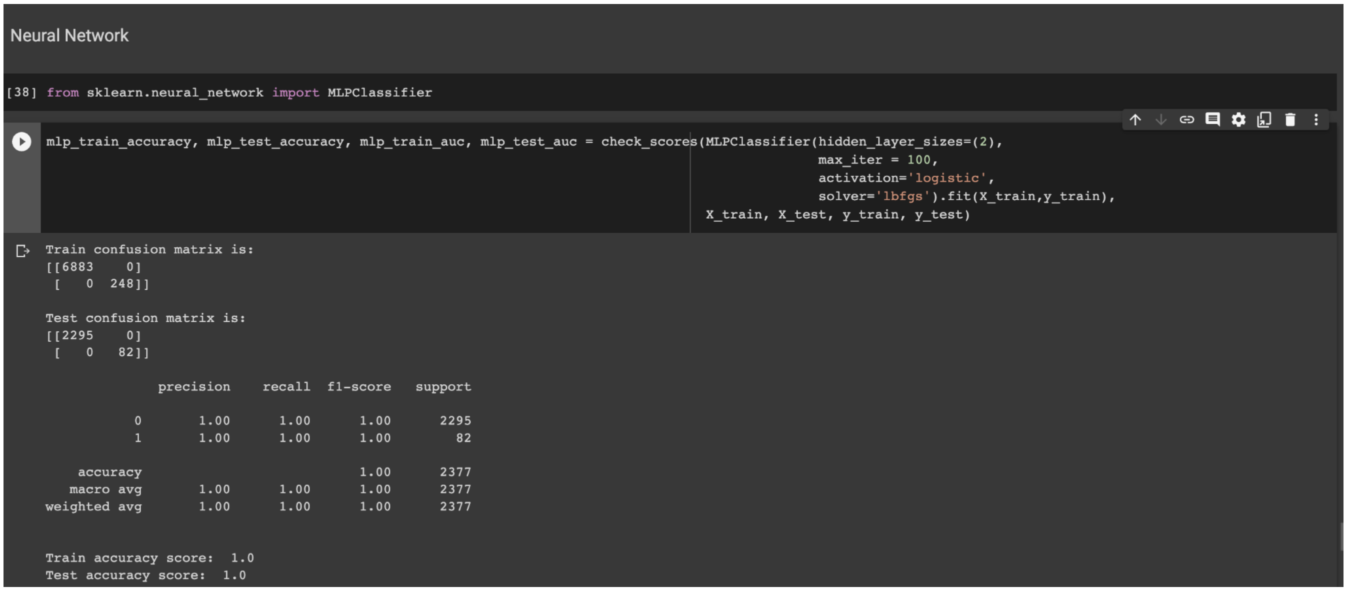
```
lda_train_accuracy, lda_test_accuracy, lda_train_auc, lda_test_auc = check_scores(LinearDiscriminantAnalysis().fit(X_train, y_train),
                                                                                   X_train, X_test, y_train, y_test)
```

```
Train confusion matrix is:
[[6882    1]
 [   4  244]]

Test confusion matrix is:
[[2189  106]
 [  61   21]]

              precision    recall  f1-score   support

           0       0.97      0.95      0.96      2295
           1       0.17      0.26      0.20        82

    accuracy                           0.93      2377
   macro avg       0.57      0.60      0.58      2377
weighted avg       0.95      0.93      0.94      2377


Train accuracy score:  0.9992988360678726
Test accuracy score:  0.9297433740008414

Train ROC-AUC score:  0.9998995303998163
Test ROC-AUC score:  0.7840533503374248
```



```
Train accuracy score:  0.9992988360678726
Test accuracy score:  0.9297433740008414

Train ROC-AUC score:  0.9998995303998163
Test ROC-AUC score:  0.7840533503374248

Area under Precision-Recall curve: 0.20095693779904306
Area under ROC-AUC: 0.18676655261640043
```

7. Neural Net Classification

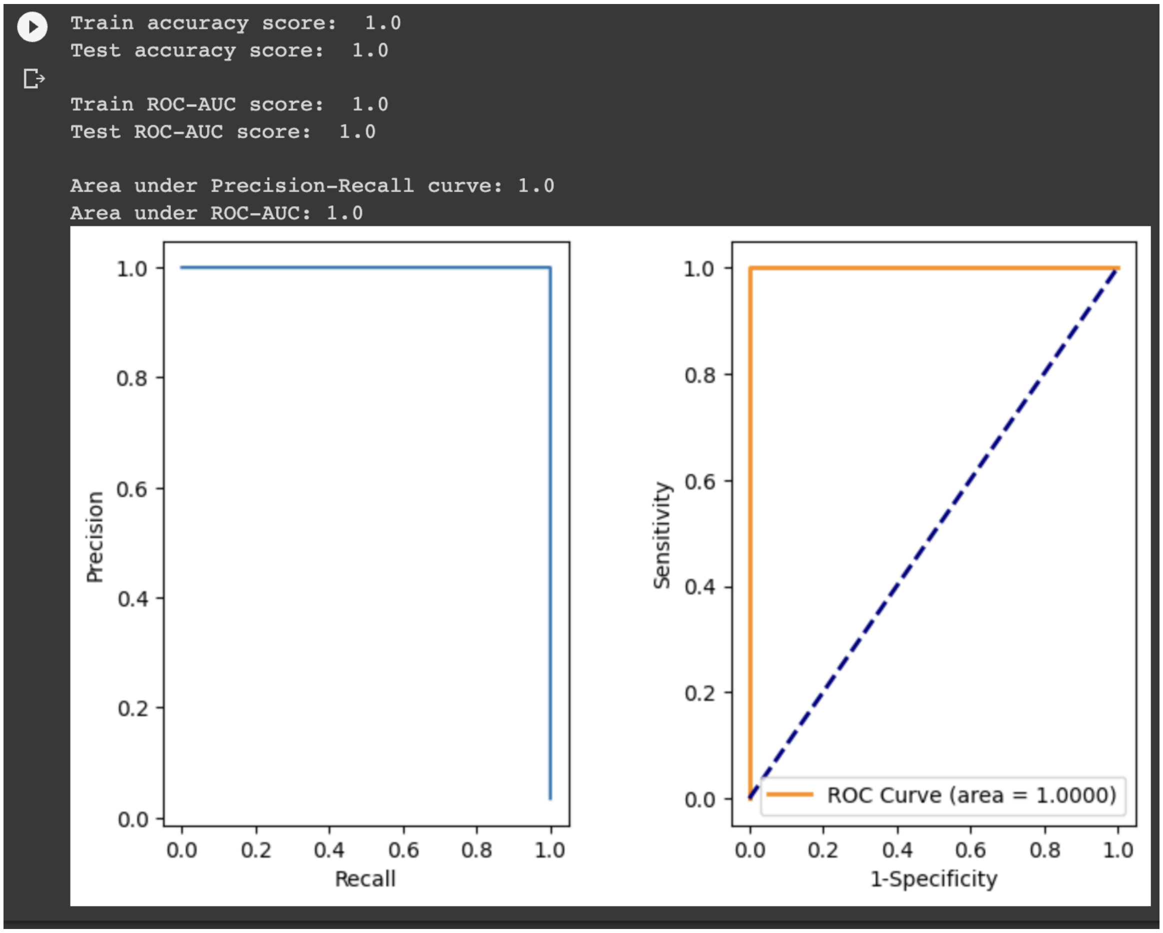


Neural Network

```
[38] from sklearn.neural_network import MLPClassifier
```

```
mlp_train_accuracy, mlp_test_accuracy, mlp_train_auc, mlp_test_auc = check_scores(MLPClassifier(hidden_layer_sizes=(2),
                                                                                       max_iter = 100,
                                                                                       activation='logistic',
                                                                                       solver='lbfgs').fit(X_train,y_train),
                                                                                X_train, X_test, y_train, y_test)
```

```
Train confusion matrix is:
[[6883    0]
 [   0  248]]

Test confusion matrix is:
[[2295    0]
 [   0   82]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2295
           1       1.00      1.00      1.00        82

    accuracy                           1.00      2377
   macro avg       1.00      1.00      1.00      2377
weighted avg       1.00      1.00      1.00      2377


Train accuracy score:  1.0
Test accuracy score:  1.0
```



```
Train accuracy score:  1.0
Test accuracy score:  1.0

Train ROC-AUC score:  1.0
Test ROC-AUC score:  1.0

Area under Precision-Recall curve: 1.0
Area under ROC-AUC: 1.0
```

The top 3 performing models for our classification problem have been **Support Vector Machines, Neural Network Classification and Logistic Regression** giving 100% accuracy. The reason for 100% accuracy is Unbalanced Dataset. The dataset contains around 2300 values for Negative Class and only around 90 for Positive Class. Another reason for it to perform so well is because of the dataframe being calculated from the sparse matrix generated from the TF-IDF vectorizer.

One improvement that can be done on this dataset is to make it more balanced by using techniques like SMOTE (Synthetic Minority Over-sampling Technique) which will create synthetic data for the under-samples class in the dataset. Since our dataset is related to text it is difficult to use techniques like SMOTE and therefore we need to find more data in terms of tweets related to US Airlines.