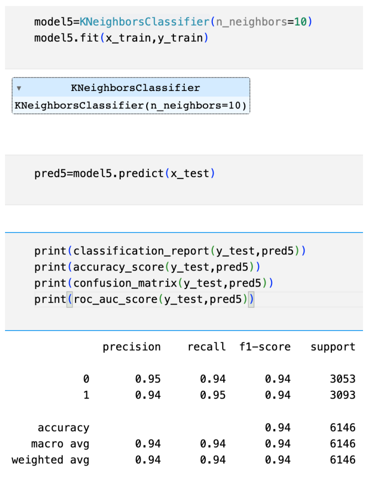
# Project Development Phase
# Model Performance Test

| | |
|---|---|
| Date | 03 November 2023 |
| Team ID | 592974 |
| Project Name | Airline Review Classification Using ML |
| Maximum Marks | 10 Marks |

**Model Performance Testing:**

| S.No. | Parameter | Values | Screenshot |
|---|---|---|---|
| 1 | Metrics | **Classification Model(KNN):**<br>Confusion Matrix -<br>[[2861, 192], [156, 2937]]<br>Accuracy Score-<br>0.943<br>Classification Report -<br>`precision recall f1-score support`<br>`0 0.95 0.94 0.94 3053`<br>`1 0.94 0.95 0.94 3093` |  |
| 2 | Tune the Model | Hyperparameter Tuning - Default<br>Validation Method - Default | |

```
model5=KNeighborsClassifier(n_neighbors=10)
model5.fit(x_train,y_train)
```

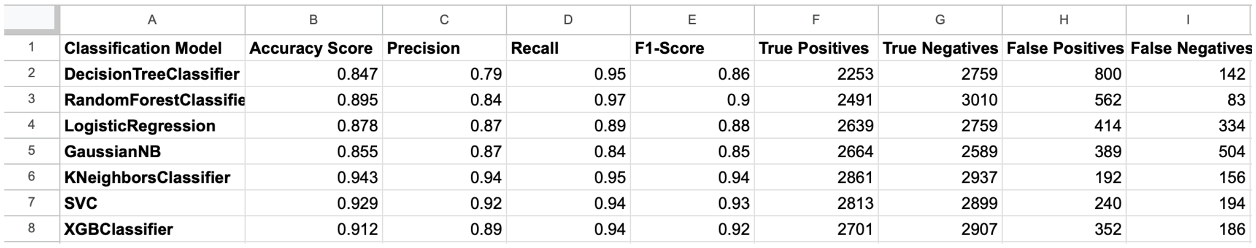
```
▾         KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

```
pred5=model5.predict(x_test)
```

```
print(classification_report(y_test,pred5))
print(accuracy_score(y_test,pred5))
print(confusion_matrix(y_test,pred5))
print(roc_auc_score(y_test,pred5))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94      3053
           1       0.94      0.95      0.94      3093

    accuracy                           0.94      6146
   macro avg       0.94      0.94      0.94      6146
weighted avg       0.94      0.94      0.94      6146
```

For all the models:

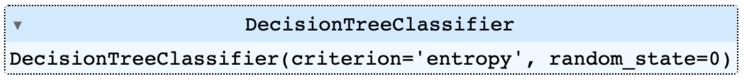


| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **Classification Model** | **Accuracy Score** | **Precision** | **Recall** | **F1-Score** | **True Positives** | **True Negatives** | **False Positives** | **False Negatives** |
| 2 | **DecisionTreeClassifier** | 0.847 | 0.79 | 0.95 | 0.86 | 2253 | 2759 | 800 | 142 |
| 3 | **RandomForestClassifie** | 0.895 | 0.84 | 0.97 | 0.9 | 2491 | 3010 | 562 | 83 |
| 4 | **LogisticRegression** | 0.878 | 0.87 | 0.89 | 0.88 | 2639 | 2759 | 414 | 334 |
| 5 | **GaussianNB** | 0.855 | 0.87 | 0.84 | 0.85 | 2664 | 2589 | 389 | 504 |
| 6 | **KNeighborsClassifier** | 0.943 | 0.94 | 0.95 | 0.94 | 2861 | 2937 | 192 | 156 |
| 7 | **SVC** | 0.929 | 0.92 | 0.94 | 0.93 | 2813 | 2899 | 240 | 194 |
| 8 | **XGBClassifier** | 0.912 | 0.89 | 0.94 | 0.92 | 2701 | 2907 | 352 | 186 |

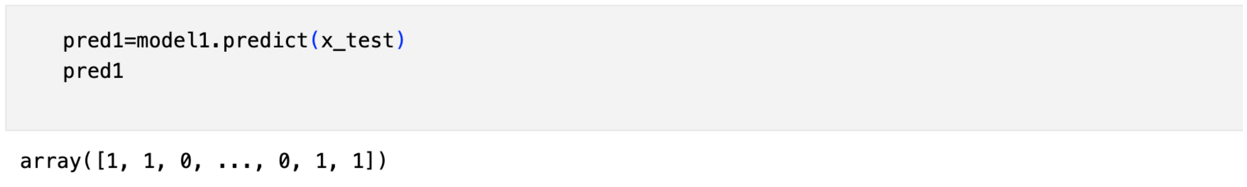
KNN performed best of all, so we used it in deployment.

Screenshots for all the models:

DecisionTreeClassifier:

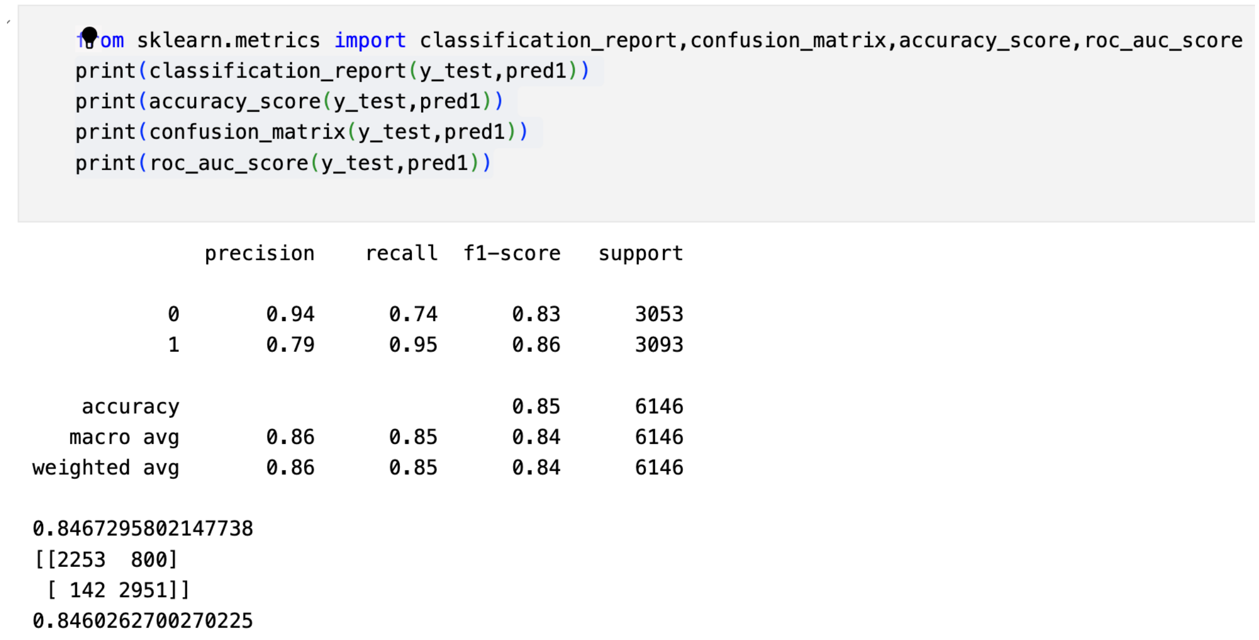


```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```



```
pred1=model1.predict(x_test)
pred1
```

```
array([1, 1, 0, ..., 0, 1, 1])
```



```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_auc_score
print(classification_report(y_test,pred1))
print(accuracy_score(y_test,pred1))
print(confusion_matrix(y_test,pred1))
print(roc_auc_score(y_test,pred1))
```

```
              precision    recall  f1-score   support

           0       0.94      0.74      0.83      3053
           1       0.79      0.95      0.86      3093

    accuracy                           0.85      6146
   macro avg       0.86      0.85      0.84      6146
weighted avg       0.86      0.85      0.84      6146

0.8467295802147738
[[2253  800]
 [ 142 2951]]
0.8460262700270225
```

RandomForestClassifier:

```
▾                         RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=20, random_state=0)
```

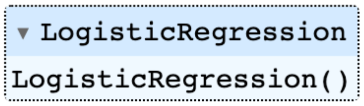
```
pred2=model2.predict(x_test)
pred2
```

```
array([1, 1, 0, ..., 0, 1, 1])
```

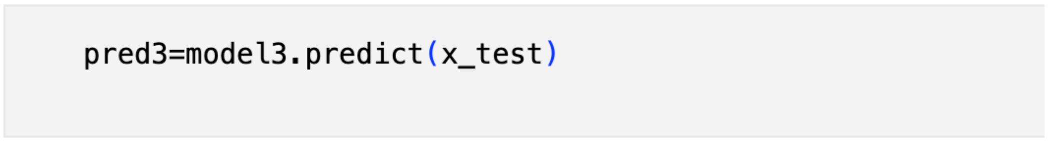
```
print(classification_report(y_test,pred2))
print(accuracy_score(y_test,pred2))
print(confusion_matrix(y_test,pred2))
print(roc_auc_score(y_test,pred2))
```

```
              precision    recall  f1-score   support

           0       0.97      0.82      0.89      3053
           1       0.84      0.97      0.90      3093

    accuracy                           0.90      6146
   macro avg       0.91      0.89      0.89      6146
weighted avg       0.90      0.90      0.89      6146

0.8950536934591604
[[2491  562]
 [  83 3010]]
0.894541990096505
```

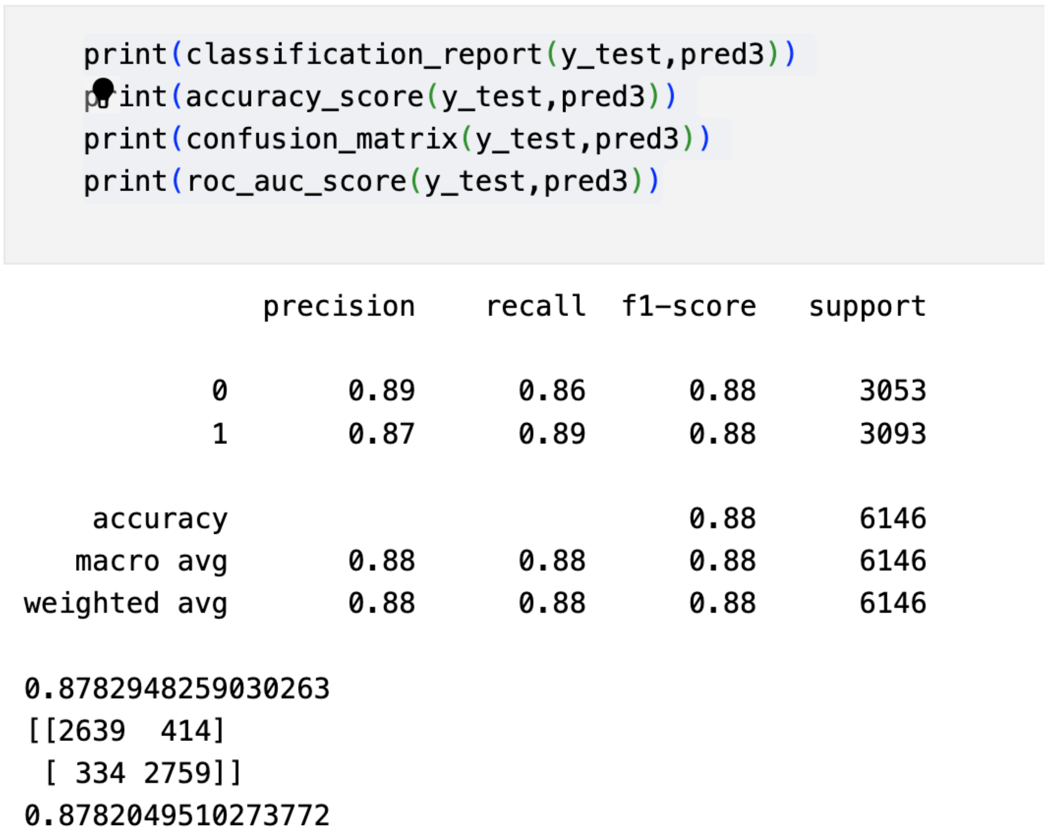
LogisticRegression:



```
▾ LogisticRegression
LogisticRegression()
```



```
pred3=model3.predict(x_test)
```



```
print(classification_report(y_test,pred3))
print(accuracy_score(y_test,pred3))
print(confusion_matrix(y_test,pred3))
print(roc_auc_score(y_test,pred3))
```

```
              precision    recall  f1-score   support

           0       0.89      0.86      0.88      3053
           1       0.87      0.89      0.88      3093

    accuracy                           0.88      6146
   macro avg       0.88      0.88      0.88      6146
weighted avg       0.88      0.88      0.88      6146

0.8782948259030263
[[2639  414]
 [ 334 2759]]
0.8782049510273772
```

GaussianNB:

```
model4=GaussianNB()
model4.fit(x_train,y_train)
```

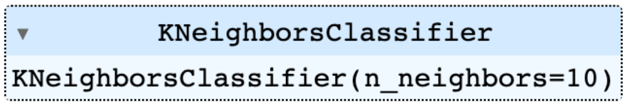
```
▾ GaussianNB
GaussianNB()
```

```
pred4=model4.predict(x_test)
```

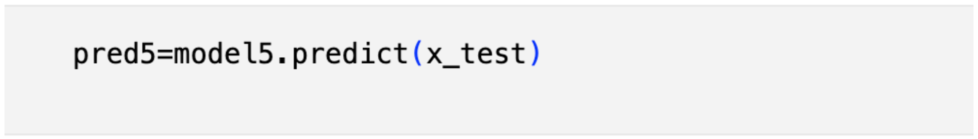
```
print(classification_report(y_test,pred4))
print(accuracy_score(y_test,pred4))
print(confusion_matrix(y_test,pred4))
print(roc_auc_score(y_test,pred4))
```

```
              precision    recall  f1-score   support

           0       0.84      0.87      0.86      3053
           1       0.87      0.84      0.85      3093

    accuracy                           0.85      6146
   macro avg       0.86      0.85      0.85      6146
weighted avg       0.86      0.85      0.85      6146

0.8547022453628376
[[2664  389]
 [ 504 2589]]
0.8548178748352337
```

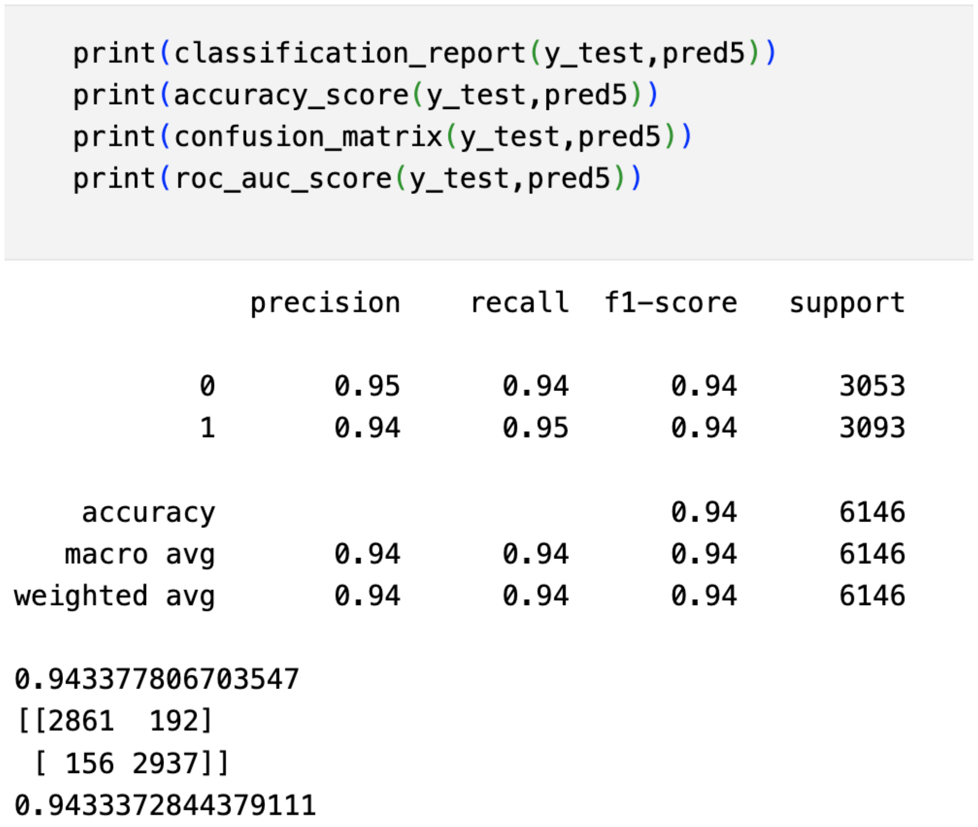
KNeighborsClassifier:



```
▾        KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```



```
pred5=model5.predict(x_test)
```



```
print(classification_report(y_test,pred5))
print(accuracy_score(y_test,pred5))
print(confusion_matrix(y_test,pred5))
print(roc_auc_score(y_test,pred5))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94      3053
           1       0.94      0.95      0.94      3093

    accuracy                           0.94      6146
   macro avg       0.94      0.94      0.94      6146
weighted avg       0.94      0.94      0.94      6146

0.943377806703547
[[2861  192]
 [ 156 2937]]
0.9433372844379111
```

SVC:

```
model6= SVC()
model6.fit(x_train,y_train)
```

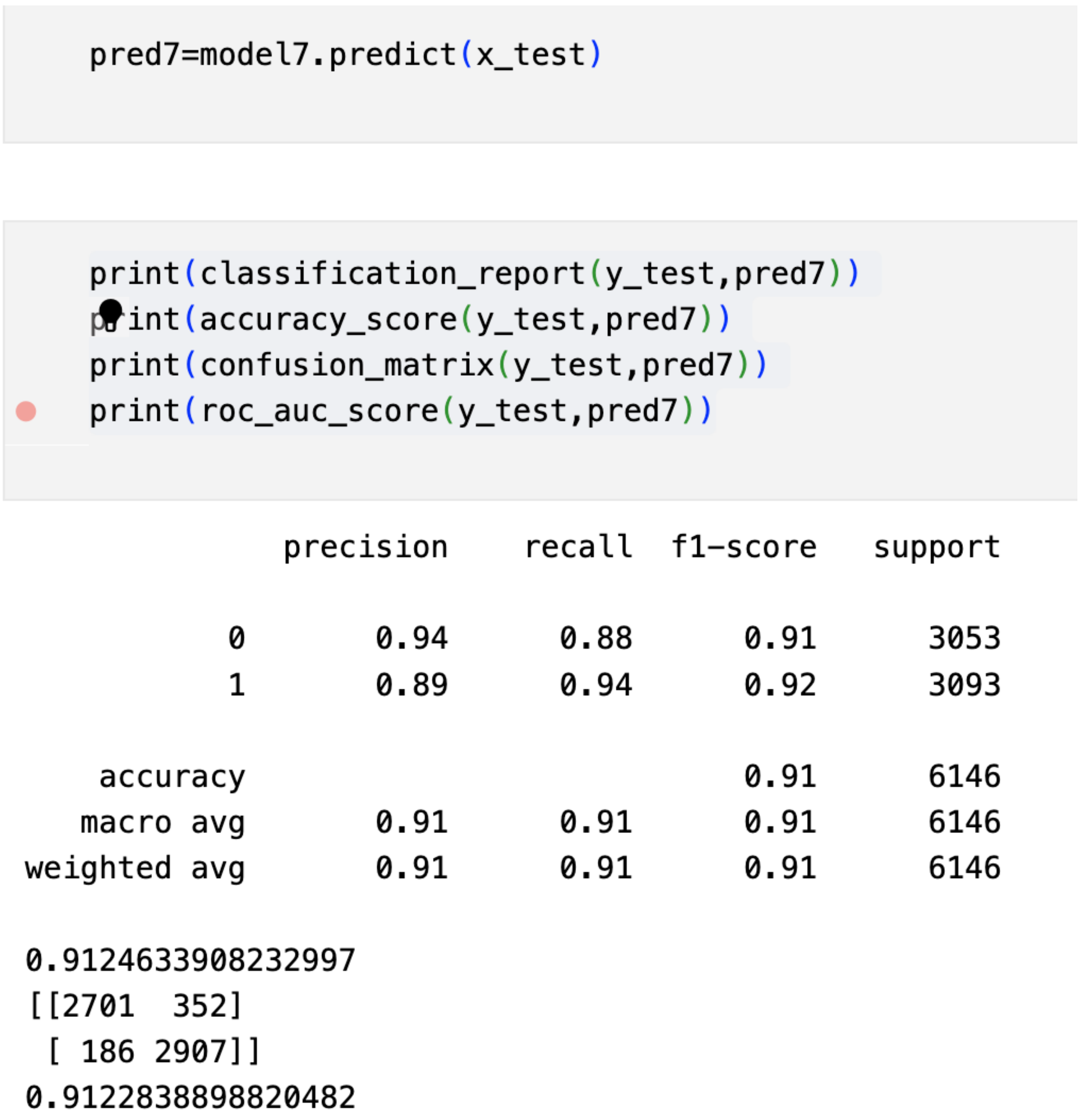
```
▾ SVC
SVC()
```

```
pred6=model6.predict(x_test)
```

```
print(classification_report(y_test,pred6))
print(accuracy_score(y_test,pred6))
print(confusion_matrix(y_test,pred6))
print(roc_auc_score(y_test,pred6))
```

```
              precision    recall  f1-score   support

           0       0.94      0.92      0.93      3053
           1       0.92      0.94      0.93      3093

    accuracy                           0.93      6146
   macro avg       0.93      0.93      0.93      6146
weighted avg       0.93      0.93      0.93      6146

0.929384965831435
[[2813  240]
 [ 194 2899]]
0.9293332608981811
```

XGBClassifier:



```
pred7=model7.predict(x_test)
```

```
print(classification_report(y_test,pred7))
print(accuracy_score(y_test,pred7))
print(confusion_matrix(y_test,pred7))
print(roc_auc_score(y_test,pred7))
```

```
              precision    recall  f1-score   support

           0       0.94      0.88      0.91      3053
           1       0.89      0.94      0.92      3093

    accuracy                           0.91      6146
   macro avg       0.91      0.91      0.91      6146
weighted avg       0.91      0.91      0.91      6146

0.9124633908232997
[[2701  352]
 [ 186 2907]]
0.9122838898820482
```