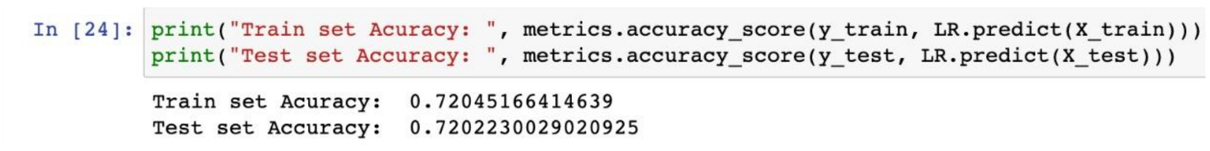
PERFORMANCE METRICS

Model: Logistic Regression performance values

There is no big variation in the training and testing accuracy. Therefore, the Logistic Regression model is not overfit or underfit.

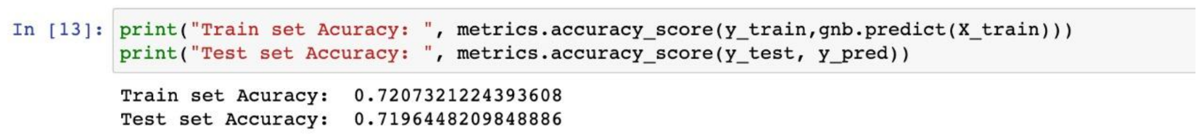


```
In [24]: print("Train set Acuracy: ", metrics.accuracy_score(y_train, LR.predict(X_train)))
         print("Test set Accuracy: ", metrics.accuracy_score(y_test, LR.predict(X_test)))

         Train set Acuracy:  0.72045166414639
         Test set Accuracy:  0.7202230029020925
```

Model: Naive Bayes performance values

There is no big variation in the training and testing accuracy

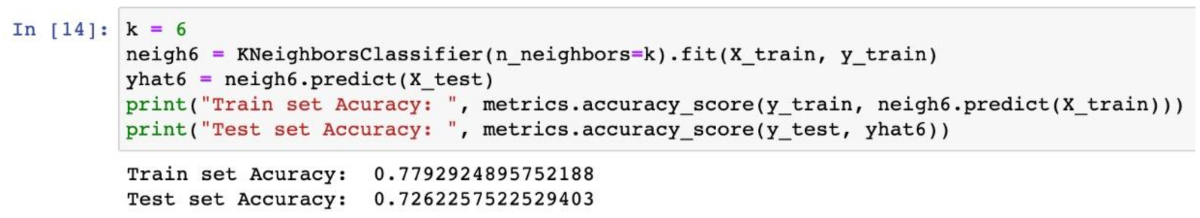


```
In [13]: print("Train set Acuracy: ", metrics.accuracy_score(y_train,gnb.predict(X_train)))
         print("Test set Accuracy: ", metrics.accuracy_score(y_test, y_pred))

         Train set Acuracy:  0.7207321224393608
         Test set Accuracy:  0.7196448209848886
```

Model: K means performance values

There is no big variation in the training and testing accuracy

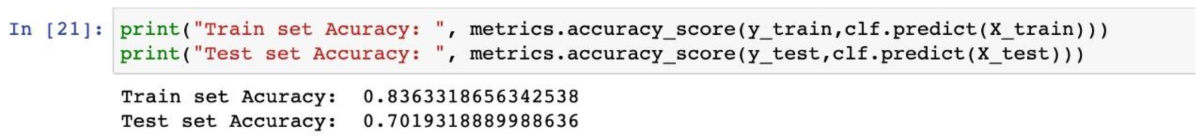


```
In [14]: k = 6
         neigh6 = KNeighborsClassifier(n_neighbors=k).fit(X_train, y_train)
         yhat6 = neigh6.predict(X_test)
         print("Train set Acuracy: ", metrics.accuracy_score(y_train, neigh6.predict(X_train)))
         print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat6))

         Train set Acuracy:  0.7792924895752188
         Test set Accuracy:  0.7262257522529403
```

Model: Random Forest performance values

There is slight variation in the training and testing accuracy



```
In [21]: print("Train set Acuracy: ", metrics.accuracy_score(y_train,clf.predict(X_train)))
         print("Test set Accuracy: ", metrics.accuracy_score(y_test,clf.predict(X_test)))

         Train set Acuracy:  0.8363318656342538
         Test set Accuracy:  0.7019318889988636
```

On comparing the four models built, based on the performance metrics it is clear that random forest gives the highest performance. Hence, that model is chosen for deployment.