Project Development Phase
Model Performance Test

Date	9th November 2023
Team ID	593197
Project Name	PayGuard Plus – An Online Payments Fraud Detector
Maximum Marks	10 Marks
Team Size	3
Member 1 – Team Lead	Akshit Bahl (21BIT0012)
Member 2	Ananya Priya (21BIT0245)
Member 3	Lakshya Mittal (21BIT0076)

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** Confusion Matrix - , Accuray Score- & Classification Report -	
2.	Tune the Model	Hyperparameter Tuning - Validation Method -	

Model chosen = **XGBoost Classifier**

- ## METRICS

1. Confusion Matrix:

```
[121] pd.crosstab(y_test1,y_test_pred5)
```

col_0	0	1
row_0		
0	108	53
1	1	237181

2. Classification Report:

```
print(classification_report(y_test1,y_test_pred5))
```

```
              precision    recall  f1-score   support

           0       0.99      0.67      0.80       161
           1       1.00      1.00      1.00    237182

    accuracy                           1.00    237343
   macro avg       1.00      0.84      0.90    237343
weighted avg       1.00      1.00      1.00    237343
```

3. Accuracy Score:

Testing accuracy: **99.977%**
Training accuracy: **99.996%**

```
[119] y_test_pred5=xgb1.predict(x_test)
      accuracy_score(y_test1,y_test_pred5)
```

```
0.9997724811770307
```

```
y_train_pred5=xgb1.predict(x_train)
accuracy_score(y_train1,y_train_pred5)
```

```
0.9999620801048756
```

- # Tuning the Model:

  a. **SMOTE:**

```
[138] from imblearn.over_sampling import SMOTE
      smote=SMOTE()
```

```
[156] x_train_smote,y_train_smote=smote.fit_resample(x_train,y_train1)
```

```
[157] xgb1.fit(x_train_smote,y_train_smote)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
[158] y_pred_xgb=xgb1.predict(x_test)
```

```
[160] accuracy_score(y_test1,y_pred_xgb)
```

```
0.9973456137320249
```

```
pd.crosstab(y_test1,y_pred_xgb)
```

col_0	0	1
row_0		
0	132	29
1	601	236581

```
print(classification_report(y_test1,y_pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.18      0.82      0.30       161
           1       1.00      1.00      1.00    237182

    accuracy                           1.00    237343
   macro avg       0.59      0.91      0.65    237343
weighted avg       1.00      1.00      1.00    237343
```

So the recall has increased with SMOTE technique but the precision and overall accuracy has been affected and has gone down.

Now, we will do hyper-parameter tuning by grid-search CV

b. HYPER_PARAMETER TUNING:

By Grid-search CV:

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
clf=GridSearchCV(rfc,param_grid=parameters,verbose=2)
```

```
[40] clf.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  15.8s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  15.3s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  16.7s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  15.9s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  14.4s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  31.6s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  33.2s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  34.2s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  30.7s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  29.1s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  15.9s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  16.4s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  16.2s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  15.6s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  14.8s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  31.3s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  30.4s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  35.1s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  31.8s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  28.8s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  14.4s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  14.1s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  15.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  14.9s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=  14.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  29.5s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  28.9s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  30.1s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  28.8s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  28.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  14.8s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  14.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  15.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  14.8s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=50; total time=  14.1s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  28.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  29.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  29.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=  26.5s
```

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

```
[42] clf.best_score_
```

```
0.9997418364499022
```

```
[43] clf.best_params_
```

```
{'max_depth': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 100}
```

```
[47] final_model=RandomForestClassifier(max_depth=None,min_samples_leaf=1,min_samples_split=2,n_estimators=100)
```

```
[48] final_model.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[49] y_pred_final=final_model.predict(x_test)
```

```
[50] accuracy_score(y_test,y_pred_final)
```

```
0.9996255504814933
```

```
pd.crosstab(y_test,y_pred_final)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	34	43
is not Fraud	3	122767

```
[52] print(classification_report(y_test,y_pred_final))
```

```
              precision    recall  f1-score   support

    is Fraud       0.92      0.44      0.60        77
is not Fraud       1.00      1.00      1.00    122770

    accuracy                           1.00    122847
   macro avg       0.96      0.72      0.80    122847
weighted avg       1.00      1.00      1.00    122847
```