Abstract

Project 1

Detecting fraud for transactions in a payment gateway

A new disruptive payment gateway start-up, 'BI Gateway', has started gaining traction due to its extremely low processing fees for handling online vendors' digital payments. This strategy has led to very low costs of acquiring new vendors.
Unfortunately, due to the cheap processing fees, the company was not able to build and deploy a robust and fast fraud detection system.

Consequently, a lot of the vendors have accumulated significant economic burden due to handling fraudulent transactions on their platforms. This has resulted in a significant number of current clients leaving BI Gateway's payment gateway platform for more expensive yet reliable payment gateway companies.

The company's data engineers curated a dataset that they believe follows the real-world distribution of transactions on their payment gateway. The company hired You and provided it with the dataset, to create a fast and robust AI based model that can detect and prevent fraudulent transactions on its payment gateway.

Tools / Skills Used:

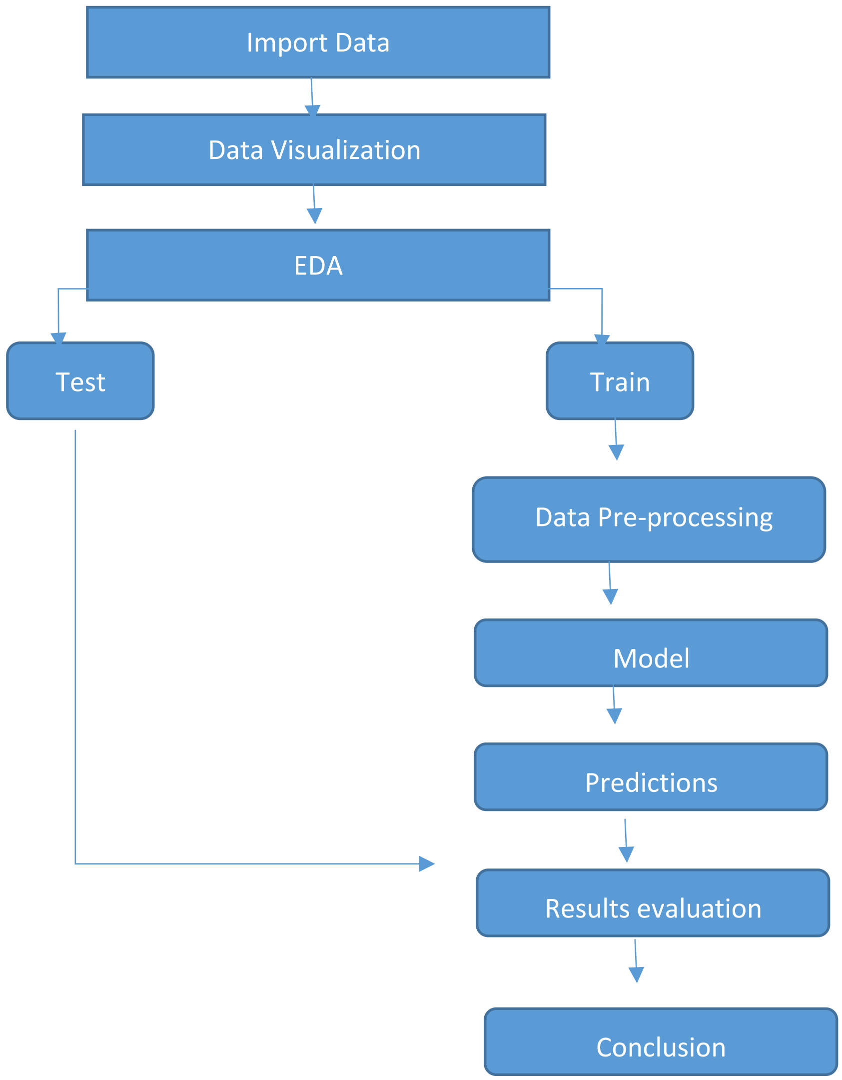
1. Python Programming
2. Jupyter Notebook
3. Pandas
4. NumPy
5. Matplotlib
6. Seaborn
7. Exploratory Data Analysis
8. Data Visualization
9. Scikitlearn
10. Machine Learning Models
11. Feature selection
12. Feature engineering
13. One hot encoding

Introduction to project 1 – Problem Statement:

Detecting fraud for transactions in a payment gateway:

Main aim of this project is to help the BI team by creating a robust a fast and AI based model that can detect and prevent fraudulent transactions on Gateway. Thus, increasing vendor retention and reducing the loss incurred by them due to the fraudulent transactions happening on the platform.

Implementation Workflow:



Modelling:

1.Logistic Regression: The logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc.

2.Random Forest: Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

3.**XG Boost :** XGBoost is a popular and efficient open-source implementation of the gradient boosted **trees** algorithm. **Gradient boosting** is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models.

Code Snippets:

Detecting fraud for transactions in a payment gateway

Importing Libraries

In [1]:
```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv
from sklearn.preprocessing import StandardScaler
#from sklearn.preprocessing import Imputer
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from matplotlib.pyplot import xticks
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=df=pd.read_csv("C:\\Users\\Darshu\\Downloads\\Fraud Detection.csv")
print("Dataset with rows{} and columns{}".format(df.shape[0],df.shape[1]))
df.head()
```

```
Dataset with rows76529 and columns11
```

Out[2]:

	transaction_number	user_id	payment_method	partner_id	partner_category	country	device_type	money_transacted	transaction_initiation	partn
0	144703125000	17539344	sbi_atm_cum_debit_card	47334	cat_1	IND_INR	android_devices	-5.0	2016-11-15 19:16:12+00:00	
1	77406814453032	24710841	e_wallet_payments	78890	cat_2	IND_INR	other_pcs	100.0	2017-01-11 09:25:33+00:00	

Univariate Analysis

In [6]:
```
df.describe()
```

Out[6]:

	transaction_number	user_id	partner_id	money_transacted	partner_pricing_category	is_fraud
count	7.652900e+04	7.652900e+04	76529.000000	76529.000000	76529.000000	76529.000000
mean	6.940200e+14	1.247483e+07	58497.189105	132.724348	2.255707	0.002012
std	7.867885e+14	1.205878e+07	36740.216787	2350.110900	0.732174	0.044814
min	8.000000e+00	1.000000e+00	7889.000000	-20000.000000	0.000000	0.000000
25%	4.387866e+13	3.515625e+06	23667.000000	-1.000000	2.000000	0.000000
50%	3.452540e+14	9.753129e+06	47334.000000	20.000000	2.000000	0.000000
75%	1.173440e+15	1.788444e+07	78890.000000	52.000000	2.000000	0.000000
max	2.784238e+15	5.592048e+07	213003.000000	197217.760000	4.000000	1.000000

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
transaction_number          0
user_id                     0
payment_method              0
partner_id                  0
partner_category            0
country                     0
device_type                 0
money_transacted            0
transaction_initiation      0
partner_pricing_category    0
is_fraud                    0
dtype: int64
```

In [8]:
```
# Understanding all columns
for col in df.select_dtypes(include='object').columns:
    print(col)
```

jupyter Fraud detection[Darshan kumar ns] Last Checkpoint: 8 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

In [15]:
```
numeric = []
categorical = []
for col in df.columns:
    if df[col].dtype=='O':
        categorical.append(col)
    else:
        numeric.append(col)
```

In [16]:
```
print("Object data type features ",categorical)
print("Numerical data type features ",numeric)
```

```
Object data type features  ['payment_method', 'partner_category', 'country', 'device_type']
Numerical data type features  ['transaction_number', 'user_id', 'partner_id', 'money_transacted', 'partner_pricing_category',
'is_fraud']
```

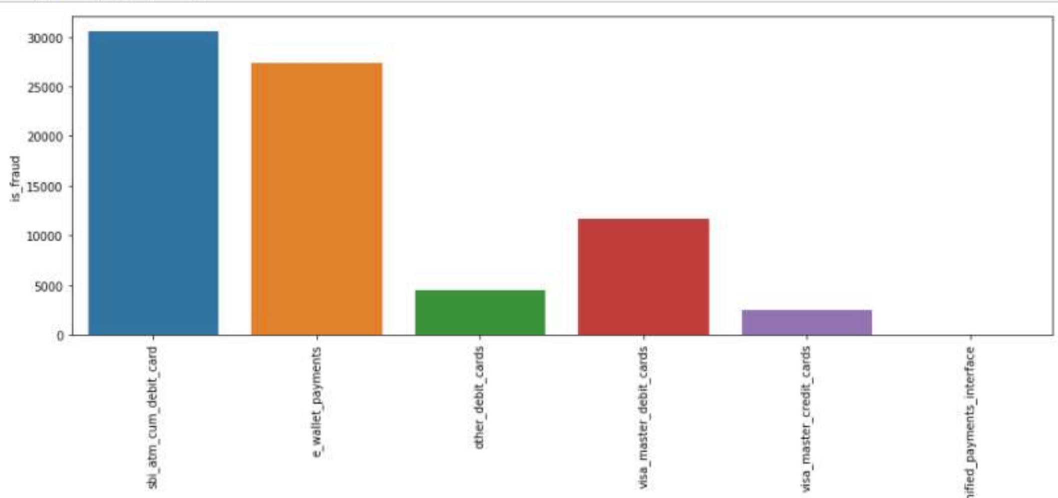
In [17]:
```
df.columns
```

Out[17]:
```
Index(['transaction_number', 'user_id', 'payment_method', 'partner_id',
       'partner_category', 'country', 'device_type', 'money_transacted',
       'partner_pricing_category', 'is_fraud'],
      dtype='object')
```

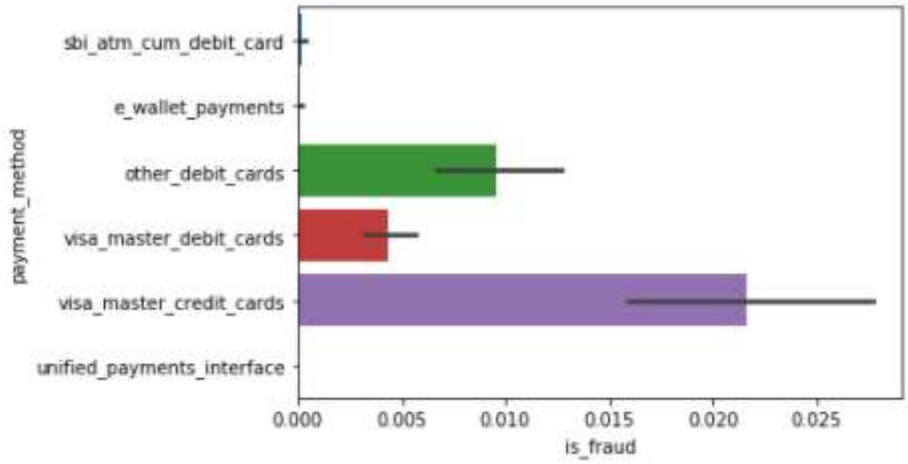
In [18]:
```
# Box plot for money trans
sns.boxplot(data=df,x='is_fraud',y='money_transacted',color='red')
plt.show()
```

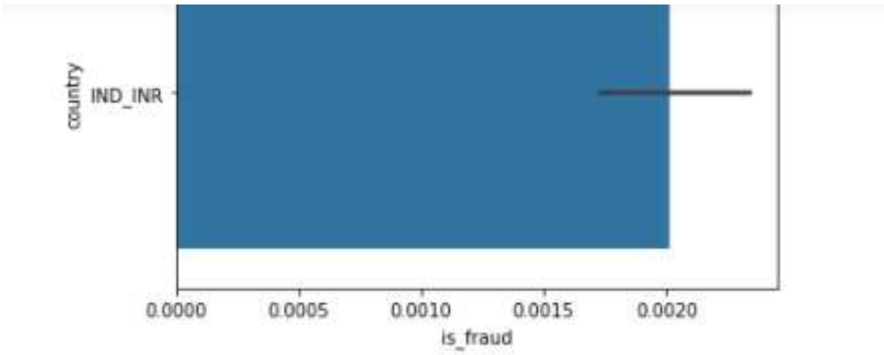
Visualization Snippets:

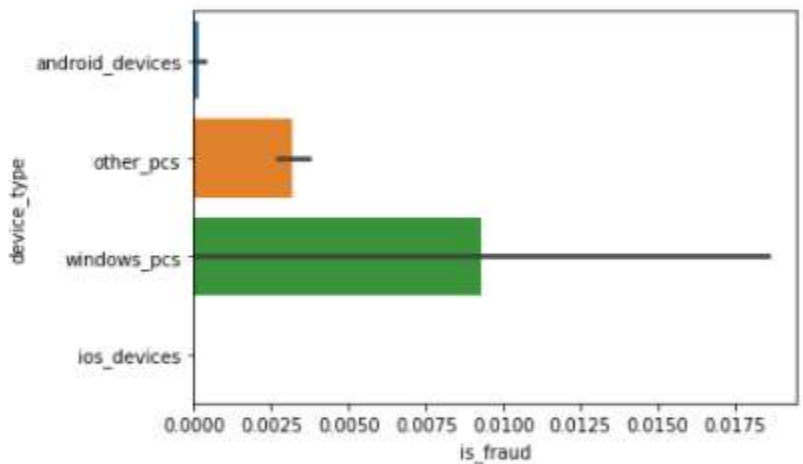
In [11]:
```
fig, ax = plt.subplots(figsize = (15,5))
plt1 = sns.countplot(df['payment_method'])
plt1.set(xlabel = 'payment_method  ', ylabel= 'is_fraud')
xticks(rotation=90)
plt.show()
plt.tight_layout()
```



```
# this picture clearley shows which holds highest farud trasaction in all segments
for i in df[categorical]:
    sns.barplot(df.is_fraud,df[i])
    plt.show()
```







Logistic Regression

In [35]:
```
X = f_data.drop("is_fraud", axis=1)
print(X.shape)
X.head()
```

(76529, 9)

Out[35]:

	payment_method	partner_category	country	device_type	transaction_number	user_id	partner_id	money_transacted	partner_pricing_category
0	2	0	0	0	144703125000	17539344	47334	-5.0	2
1	0	1	0	2	77406814453032	24710841	78890	100.0	2
2	0	1	0	2	308929485482801	24265476	78890	50.0	2
3	1	2	0	2	665270027747073	10240000	102557	1000.0	2
4	1	0	0	2	38276160171101	5880625	118335	200.0	2

In [36]:
```
y= f_data[['is_fraud']]
print(y.shape)
y.head()
```

(76529, 1)

Out[36]:

	is_fraud
0	0

Predictive model : RandomForest

In [41]:
```
# impoting random forest classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(max_depth=5,random_state=143,max_leaf_nodes=50)
```

Splitting into Train and Test

In [42]:
```
X = f_data.drop("is_fraud", axis=1)
print(X.shape)
X.head()
```

```
(76529, 9)
```

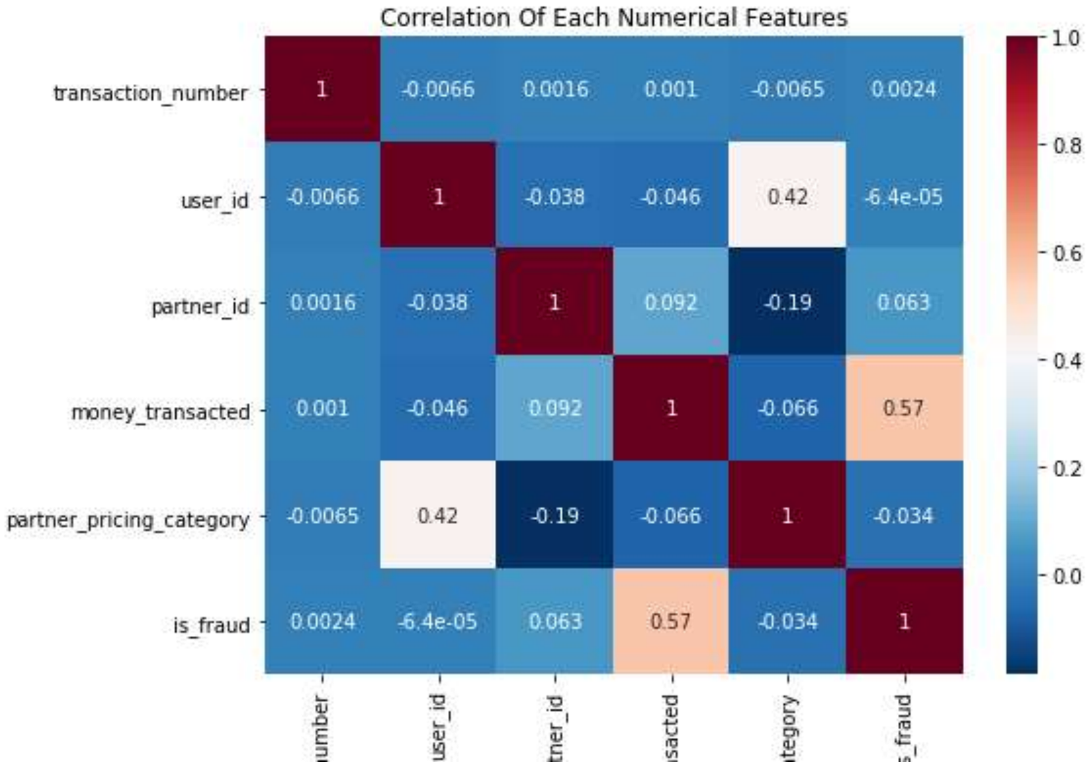
Out[42]:

	payment_method	partner_category	country	device_type	transaction_number	user_id	partner_id	money_transacted	partner_pricing_category
0	2	0	0	0	144703125000	17539344	47334	-5.0	2
1	0	1	0	2	77406814453032	24710841	78890	100.0	2
2	0	1	0	2	308929485482801	24265476	78890	50.0	2
3	1	2	0	2	665270027747073	10240000	102557	1000.0	2
4	1	0	0	2	38276160171101	5880625	118335	200.0	2

In [43]:
```
y= f_data[['is_fraud']]
print(y.shape)
y.head()
```

```
(76529, 1)
```

In [24]:
```python
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(),annot=True,cmap='RdBu_r')
plt.title("Correlation Of Each Numerical Features")
plt.show()
```



In [54]:

```
rfc_params = grid_search.best_params_
print(rfc_params)

rfc_model = RandomForestClassifier()

rfc_model.set_params(**rfc_params)
rfc_model.fit(X_train,y_train)

y_train_pred = rfc_model.predict(X_train)
y_test_pred = rfc_model.predict(X_test)


#### VALIDATION #######

#validating on train
print('recall_score for train',recall_score(y_train,y_train_pred))
print('precision_score for train',precision_score(y_train,y_train_pred))
print('f1_score for train',f1_score(y_train,y_train_pred))

# validate on test
print('recall_score for test',recall_score(y_test,y_test_pred))
print('precision_score for test',precision_score(y_test,y_test_pred))
print('f1_score for test',f1_score(y_test,y_test_pred))
```

```
{'max_depth': 9, 'max_features': 7, 'n_estimators': 50}
recall_score for train 0.9916666666666667
precision_score for train 1.0
f1_score for train 0.99581589958159
recall_score for test 0.7941176470588235
precision_score for test 0.9
f1_score for test 0.84375
```

In [54]:

```
rfc_params = grid_search.best_params_
print(rfc_params)

rfc_model = RandomForestClassifier()

rfc_model.set_params(**rfc_params)
rfc_model.fit(X_train,y_train)

y_train_pred = rfc_model.predict(X_train)
y_test_pred = rfc_model.predict(X_test)


#### VALIDATION #######

#validating on train
print('recall_score for train',recall_score(y_train,y_train_pred))
print('precision_score for train',precision_score(y_train,y_train_pred))
print('f1_score for train',f1_score(y_train,y_train_pred))

# validate on test
print('recall_score for test',recall_score(y_test,y_test_pred))
print('precision_score for test',precision_score(y_test,y_test_pred))
print('f1_score for test',f1_score(y_test,y_test_pred))
```

```
{'max_depth': 9, 'max_features': 7, 'n_estimators': 50}
recall_score for train 0.9916666666666667
precision_score for train 1.0
f1_score for train 0.99581589958159
recall_score for test 0.7941176470588235
precision_score for test 0.9
f1_score for test 0.84375
```

XGBOOST

In [73]:

```
# importing xgboost
import xgboost as xgb
from xgboost import XGBClassifier
XGBoost_CLF = xgb.XGBClassifier(max_depth=6, learning_rate=0.05, n_estimators=400,
                                objective="binary:hinge", booster='gbtree',
                                n_jobs=-1, nthread=None, gamma=0, min_child_weight=1, max_delta_step=0,
                                subsample=1, colsample_bytree=1, colsample_bylevel=1, reg_alpha=0, reg_lambda=1,
                                base_score=0.5, random_state=42)

XGBoost_CLF.fit(X_train,y_train)

y_pred = XGBoost_CLF.predict(X_test)

print("Classification Report for XGBoost: \n", classification_report(y_test, y_pred))
print("Confusion Matrix of XGBoost: \n", confusion_matrix(y_test,y_pred))
```

```
Classification Report for XGBoost:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     15272
           1       0.93      0.82      0.87        34

    accuracy                           1.00     15306
   macro avg       0.97      0.91      0.94     15306
weighted avg       1.00      1.00      1.00     15306

Confusion Matrix of XGBoost:
 [[15270     2]
 [    6    28]]
```

Conclusion/ Results:

From visualization:

- Fraud has happened in high transactions amount
- Category 8> Category 3> Category 1 partner category had higher fraud percentage.
- Visa master credit card> visa master debit card> other debit cards payment method had higher fraud percentage amount wise. Therefore, BI Gateway should increase security on these payment methods.
- Windows Pc and other Pcs has more fraud percentage.

From Models:

- Both Logistic regression and precision did poor in terms of precision however, recall was better for both of them. Post grid search, the precision increased for random forest.
- Here, both precision and recall are important matrix since we need to identify the number of fraud detected correctly/incorrectly from total fraud than accuracy. Accuracy is good for both models however; we are not weighing it as important matrix here.
- XGBoost did decent in terms of both precision and recall of **85% and 87%** respectively. Therefore, will be choice of model in terms of fraud detection for us.

Future Scope

- The model can be trained with datasets from other payment gateways, e-commerce, Banks etc. to make it more robustness and applicable on large variety of datasets.
- The economic impact pre- and post-implementation of the model needs to be assessed to determine the effectiveness of the model in the real- scenario.