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

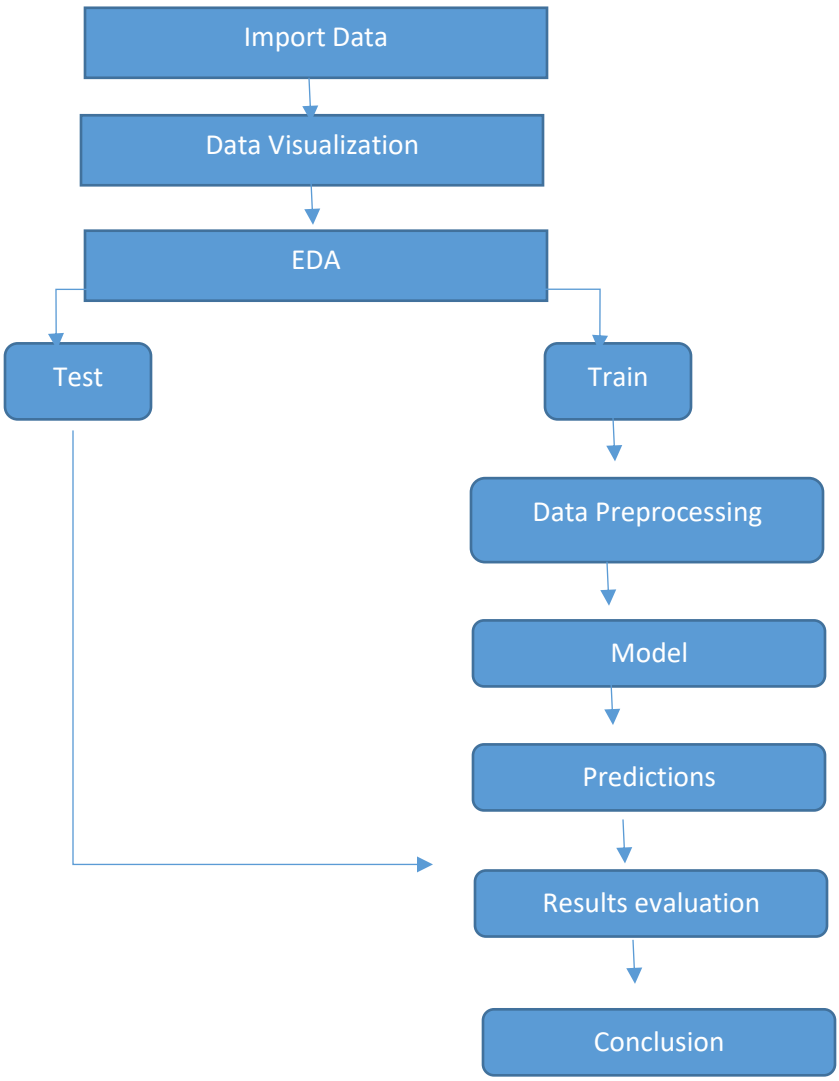
1. Python Programming
2. Jupyter Notebook
3. Pandas
4. Numpy
5. Matplotlib
6. Seaborn
7. Exploratory Data Analysis
8. Data Visualization
9. Scikitlearn
10. Machine Learning Models
11. Feature selection
12. Feature engineering
13. Standard Scaling
14. SMOTE

Introduction to project 1 – Problem Statement:

Detecting fraud for transactions in a payment gateway:

Help create a robust a fast and robust AI based model that can detect and prevent fraudulent transactions on BI Gateway. Thus increasing vendor retention and reducing the loss incurred by them due to the fraudulent transactions happening on the platform

Implementation Workflow:



Modelling:

3 models were used and then the result was compared on classification matrix.

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. XGBoost : XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

Code Snippets:

Importing libraries

```
2]: import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import os

    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from scipy.special import expit
    from sklearn.metrics import confusion_matrix
    from sklearn import svm
    import datetime
```

Importing the data, null ,describe and conversion of datatypes

```
os.chdir('C:/Users/soumya/Desktop/BI/Soumya - Projects')
df = pd.read_csv('Fraud Detection.csv')
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76529 entries, 0 to 76528
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   transaction_number       76529 non-null  int64
 1   user_id                  76529 non-null  int64
 2   payment_method           76529 non-null  object
 3   partner_id               76529 non-null  int64
 4   partner_category         76529 non-null  object
 5   country                  76529 non-null  object
 6   device_type              76529 non-null  object
 7   money_transacted         76529 non-null  float64
 8   transaction_initiation   76529 non-null  object
 9   partner_pricing_category  76529 non-null  int64
 10  is_fraud                 76529 non-null  int64
dtypes: float64(1), int64(5), object(5)
memory usage: 6.4+ MB
None
```

```
: df['partner_id'] = df['partner_id'].astype('category')
  df['partner_pricing_category'] = df['partner_pricing_category'].astype('category')
  df['user_id'] = df['user_id'].astype('object')
  df['transaction_number'] = df['transaction_number'].astype('object')
  df['transaction_initiation'] = pd.to_datetime(df['transaction_initiation'], infer_datetime_format=True)
```

```
: df.describe()
```

:

	money_transacted	is_fraud
count	76529.000000	76529.000000
mean	132.724348	0.002012
std	2350.110900	0.044814
min	-20000.000000	0.000000
25%	-1.000000	0.000000
50%	20.000000	0.000000
75%	52.000000	0.000000
max	197217.760000	1.000000

```
payment_df = pd.DataFrame ({'pay_method': df.groupby(['payment_method', 'Fraud_Label'] ) ["transaction_number"].nunique()})
payment_pivot = pd.pivot_table(payment_df, values = 'pay_method', index = ['payment_method'], columns = ['Fraud_Label']).reset_ir
payment_pivot['fraud'].fillna(0, inplace = True)
payment_pivot['payment_percentage']= (payment_pivot['fraud']/(payment_pivot['fraud']+payment_pivot['Not_fraud']))*100
payment_pivot['total_trans'] = (payment_pivot['fraud']+ payment_pivot['Not_fraud'])
payment_pivot['percent_trans'] = (payment_pivot['total_trans']/ payment_pivot['total_trans'].sum())*100

payment_pivot['count_of_fraud'] = (payment_pivot['percent_trans']*payment_pivot['payment_percentage'])/100


payment_pivot = payment_pivot.reset_index().rename_axis(None, axis=1)

payment_pivot
```

	index	payment_method	Not_fraud	fraud	payment_percentage	total_trans	percent_trans	count_of_fraud
0	0	e_wallet_payments	27382.0	2.0	0.007304	27384.0	35.782514	0.002613
1	1	other_debit_cards	4452.0	43.0	0.956618	4495.0	5.873590	0.056188
2	2	sbi_atm_cum_debit_card	30533.0	5.0	0.016373	30538.0	39.903827	0.006533
3	3	unified_payments_interface	15.0	0.0	0.000000	15.0	0.019600	0.000000
4	4	visa_master_credit_cards	2401.0	53.0	2.159739	2454.0	3.206628	0.069255
5	5	visa_master_debit_cards	11592.0	51.0	0.438031	11643.0	15.213841	0.066641

Separating out numerical and categorical attributes

[31]:
```
cat_attr = list(df.select_dtypes(include=['category','object']).columns)
num_attr = list(df.select_dtypes(include=['number']).columns)

print(cat_attr)
print(num_attr)
```

```
['transaction_number', 'user_id', 'payment_method', 'partner_id', 'partner_category', 'country', 'device_type', 'partner_pricin
g_category', 'Hours', 'day', 'Fraud_Label']
['money_transacted', 'is_fraud']
```

Removing irrelevant columns and splitting into Train and Test

[32]:
```
from sklearn.model_selection import train_test_split
X = df.drop(['is_fraud','transaction_number','user_id','transaction_initiation','country','Fraud_Label'], axis=1)
y = df['is_fraud']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=23, stratify = y)

print(y_train.value_counts(normalize=True))
print(y_test.value_counts(normalize=True))
```

```
0    0.997996
1    0.002004
Name: is_fraud, dtype: float64
0    0.997962
1    0.002038
Name: is_fraud, dtype: float64
```

OHE & Standard scaling

]:
```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
ohe = OneHotEncoder(handle_unknown='error')

ohe.fit(X_train[cat_attr])

columns_ohe = list(ohe.get_feature_names(cat_attr))
print(columns_ohe)

X_train_cat = ohe.transform(X_train[cat_attr])
X_test_cat = ohe.transform(X_test[cat_attr])

X_train_cat = pd.DataFrame(X_train_cat.todense(), columns=columns_ohe)
X_test_cat = pd.DataFrame(X_test_cat.todense(), columns=columns_ohe)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train[num_attr])

X_train_std = pd.DataFrame(scaler.transform(X_train[num_attr]), columns=num_attr)
X_test_std = pd.DataFrame(scaler.transform(X_test[num_attr]), columns=num_attr)
```

```
X_train_preprocessed = pd.concat([X_train_std, X_train_cat], axis=1)
X_test_preprocessed = pd.concat([X_test_std, X_test_cat], axis=1)
X_train_preprocessed.head()
```

OVERSAMPLING with SMOTE

```
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_sample(X_train_preprocessed, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
After OverSampling, the shape of train_X: (114562, 102)
After OverSampling, the shape of train_y: (114562,)

After OverSampling, counts of label '1': 57281
After OverSampling, counts of label '0': 57281
```

LOGISTIC REGRESSION ¶

```
from sklearn.linear_model import LogisticRegression

lg = LogisticRegression()

lg.fit(X_train_res,y_train_res)

y_train_pred = lg.predict(X_train_res)
y_test_pred = lg.predict(X_test_preprocessed)


print(y_train_pred)
print(y_test_pred)
```

```
[0 0 0 ... 1 1 1]
[0 0 0 ... 0 0 0]
```

```
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     19094
           1       0.12      0.90      0.21        39

    accuracy                           0.99     19133
   macro avg       0.56      0.94      0.60     19133
weighted avg       1.00      0.99      0.99     19133
```

```
]: ## Feature Importance
   from sklearn.feature_selection import SelectFromModel
   smf = SelectFromModel(lg)
   smf.fit(X_train_preprocessed,y_train)
   features = smf.get_support()
   feature_name = X_train_preprocessed.columns[features]
   feature_name
```

```
]: Index(['money_transacted', 'payment_method_e_wallet_payments',
         'payment_method_sbi_atm_cum_debit_card',
         'payment_method_visa_master_credit_cards', 'partner_id_23667',
         'partner_id_39445', 'partner_id_47334', 'partner_id_71001',
         'partner_id_78890', 'partner_id_102557', 'partner_id_118335',
         'partner_id_173558', 'partner_category_cat_2', 'partner_category_cat_3',
         'partner_category_cat_5', 'partner_category_cat_8',
         'device_type_windows_pcs', 'partner_pricing_category_1',
         'partner_pricing_category_4', 'Hours_0', 'Hours_4', 'Hours_6',
         'Hours_10', 'Hours_12', 'Hours_13', 'Hours_21', 'day_1', 'day_3',
         'day_4', 'day_5', 'day_7', 'day_9', 'day_10', 'day_13', 'day_16',
         'day_17', 'day_18', 'day_20', 'day_22', 'day_23', 'day_24', 'day_25',
         'day_28'],
        dtype='object')
```

RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=10, max_features=3,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=50,
                       n_jobs=None, oob_score=True, random_state=None,
                       verbose=0, warm_start=False)
rf_model.fit(X_train_res, y_train_res)

y_train_pred = rf_model.predict(X_train_res)
y_test_pred = rf_model.predict(X_test_preprocessed)
print(y_train_pred)
print(y_test_pred)
```

```
[0 1 0 ... 1 1 1]
[0 0 0 ... 0 0 0]
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_test_pred)
print(confusion_matrix)
print("Classification Report for random: \n", classification_report(y_test, y_test_pred))
```

```
[[17760  1334]
 [    6    33]]
Classification Report for random:
              precision    recall  f1-score   support

           0       1.00      0.93      0.96     19094
           1       0.02      0.85      0.05        39

    accuracy                           0.93     19133
   macro avg       0.51      0.89      0.51     19133
weighted avg       1.00      0.93      0.96     19133
```

```
from sklearn.model_selection import GridSearchCV

grid_search = GridSearchCV(estimator=RandomForestClassifier(oob_score = True),
            param_grid = {'n_estimators' : [10, 20, 30, 40, 50],
                          'max_depth' : [2, 3, 4, 5, 7, 8, 9, 10],
                          'max_features' : [2, 3, 4, 5, 6, 7]})

grid_search.fit(X_train_preprocessed, y_train)

grid_search.best_estimator_
```

```
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:540: UserWarning: Some inputs do not have OOB score
s. This probably means too few trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:544: RuntimeWarning: invalid value encountered in tr
ue_divide
  decision = (predictions[k] /
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:540: UserWarning: Some inputs do not have OOB score
s. This probably means too few trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:544: RuntimeWarning: invalid value encountered in tr
ue_divide
  decision = (predictions[k] /
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:540: UserWarning: Some inputs do not have OOB score
s. This probably means too few trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:544: RuntimeWarning: invalid value encountered in tr
ue_divide
  decision = (predictions[k] /
```

```
RandomForestClassifier(max_depth=9, max_features=7, n_estimators=10,
                       oob_score=True)
```

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=10, max_features=7,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=True, random_state=None,
                       verbose=0, warm_start=False)
rf_model.fit(X_train_res, y_train_res)

y_train_pred = rf_model.predict(X_train_res)
y_test_pred = rf_model.predict(X_test_preprocessed)
print(y_train_pred)
print(y_test_pred)
```

```
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:540: UserWarning: Some inputs do not have OOB scores. T
his probably means too few trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\soumya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:544: RuntimeWarning: invalid value encountered in true_
divide
  decision = (predictions[k] /
```

```
[0 0 0 ... 1 1 1]
[0 0 0 ... 0 0 0]
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_test_pred)
print(confusion_matrix)
print("Classification Report for random: \n", classification_report(y_test, y_test_pred))
```

```
[[18881   213]
 [    5    34]]
Classification Report for random:
               precision    recall  f1-score   support

           0       1.00      0.99      0.99     19094
           1       0.14      0.87      0.24        39

    accuracy                           0.99     19133
   macro avg       0.57      0.93      0.62     19133
weighted avg       1.00      0.99      0.99     19133
```

```
: from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_test_pred)
print(confusion_matrix)
print("Classification Report for random: \n", classification_report(y_test, y_test_pred))
```

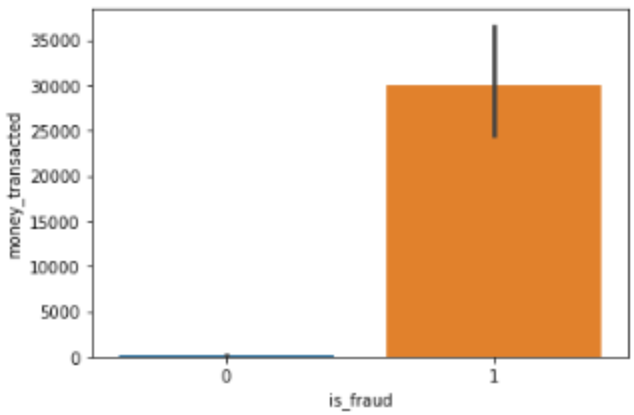
```
[[18881   213]
 [    5    34]]
Classification Report for random:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     19094
           1       0.14      0.87      0.24        39

    accuracy                           0.99     19133
   macro avg       0.57      0.93      0.62     19133
weighted avg       1.00      0.99      0.99     19133
```

XGBoost

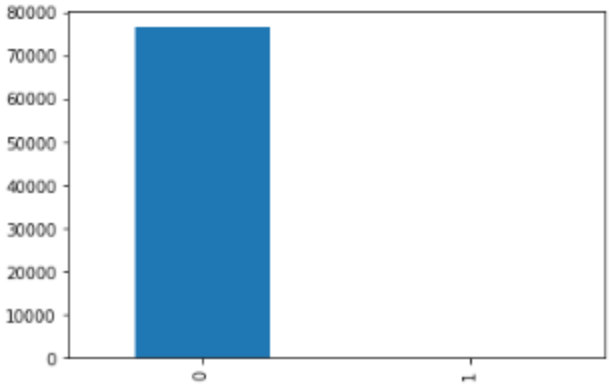
```
1]: import xgboost as xgb
from xgboost import XGBClassifier
XGBoost_CLF = xgb.XGBClassifier(max_depth=6, learning_rate=0.05, n_estimators=400,
                                objective="binary:hinge", booster='gbtree',
                                n_jobs=-1, nthread=None, gamma=0, min_child_weight=1, 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_res,y_train_res)

y_pred = XGBoost_CLF.predict(X_test_preprocessed)

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     19094
           1       0.89      0.85      0.87        39

    accuracy                           1.00     19133
   macro avg       0.95      0.92      0.93     19133
weighted avg       1.00      1.00      1.00     19133

Confusion Matrix of XGBoost:
 [[19090     4]
 [    6    33]]
```

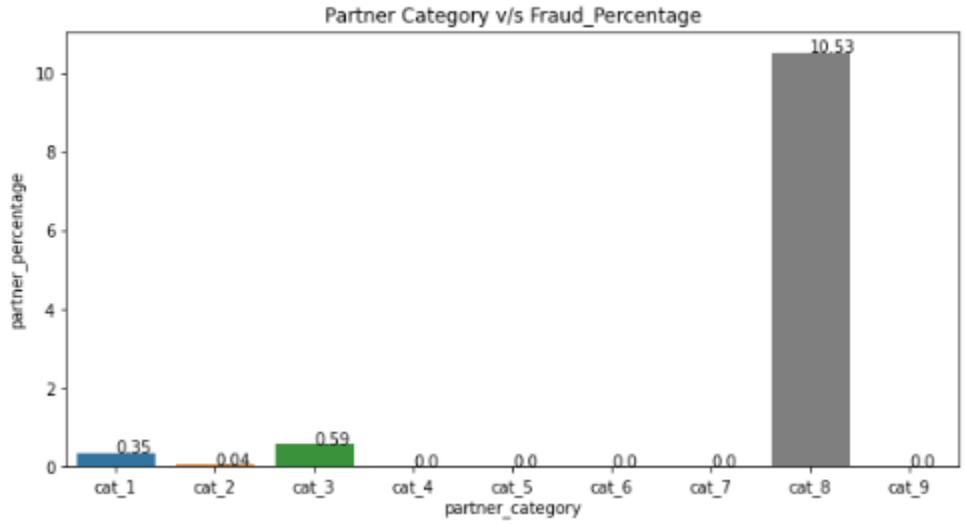
Visualization Snippets:

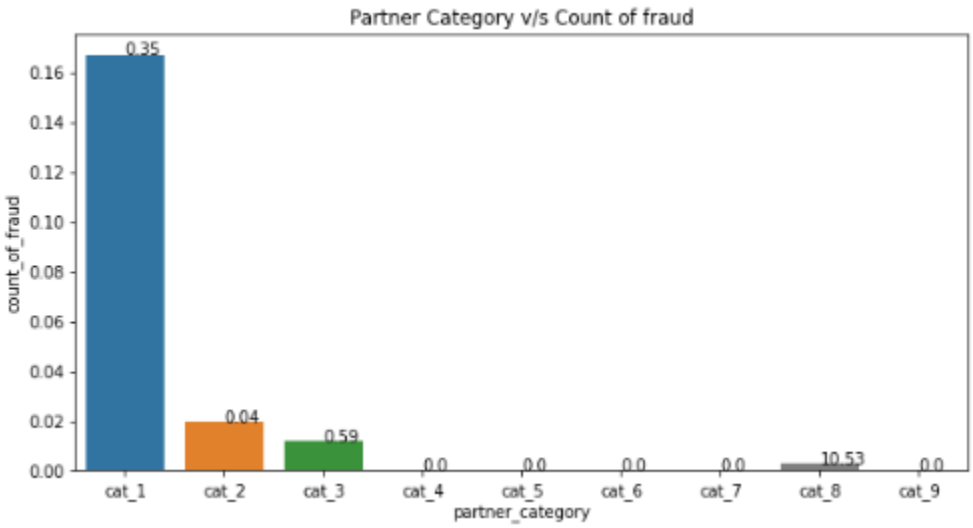


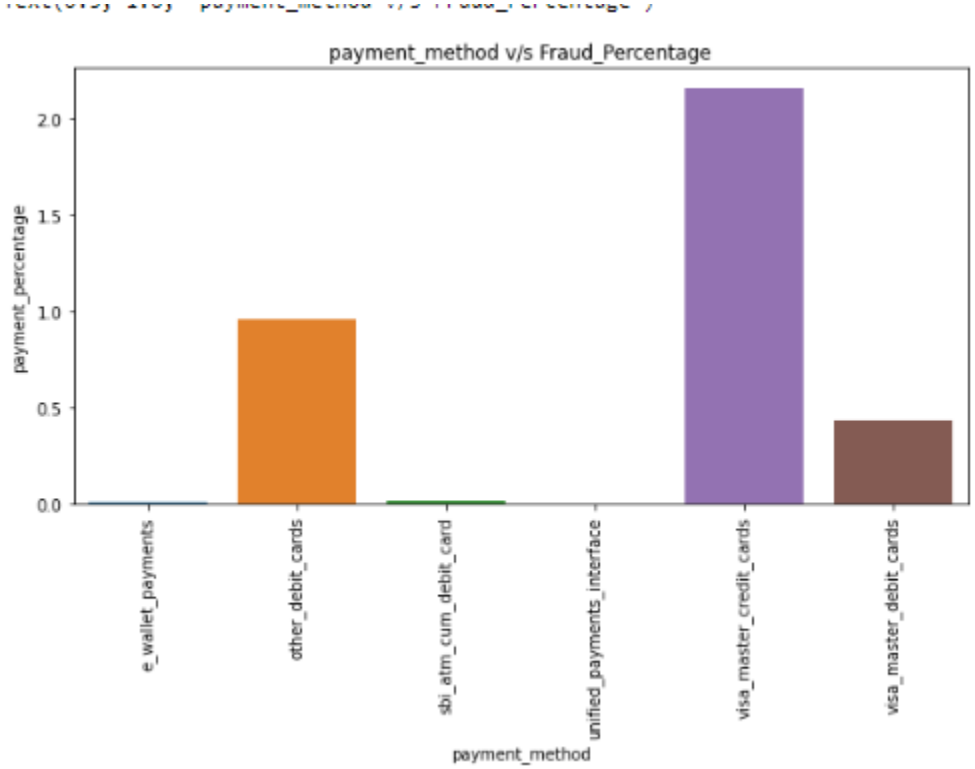

```
AxesSubplot(0.125,0.125;0.775x0.755)
Total number of fraud 154
```

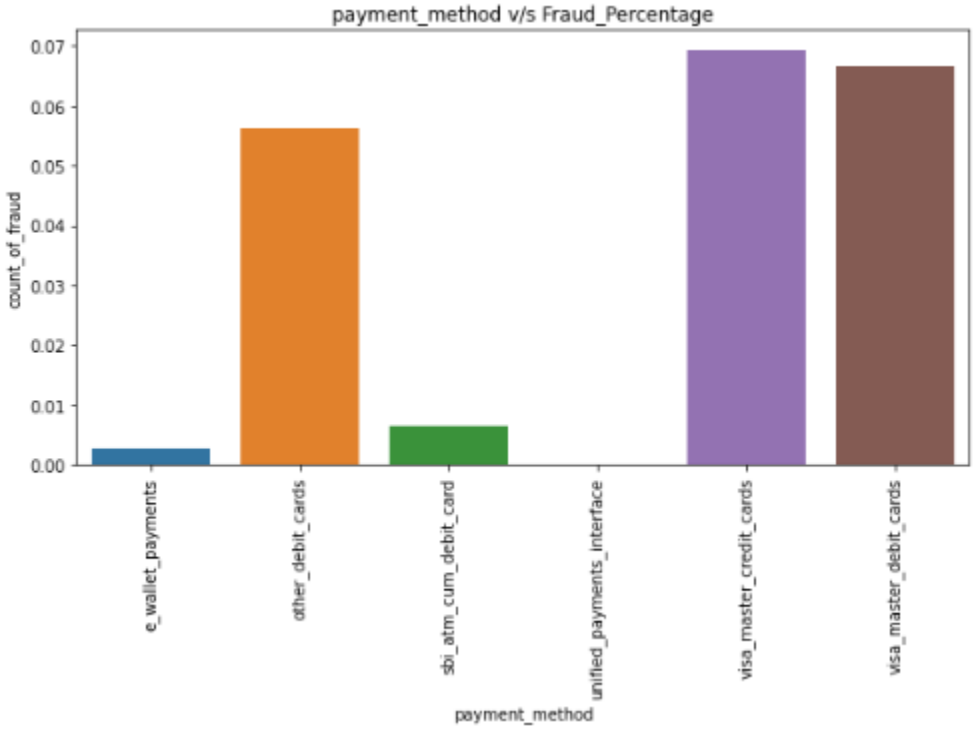


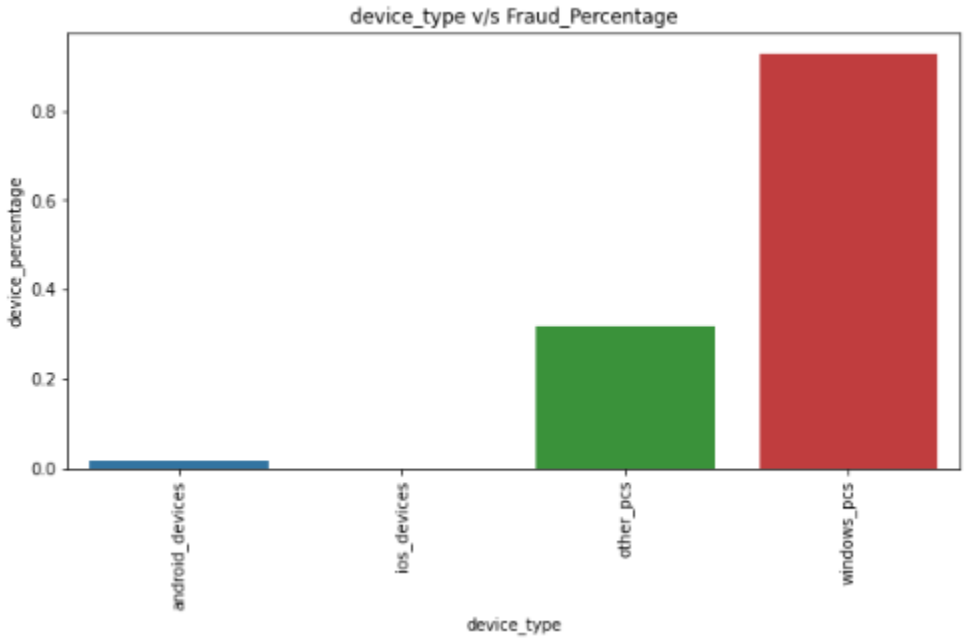
```
: Text(0.5, 1.0, 'Partner Category v/s Fraud_Percentage')
```

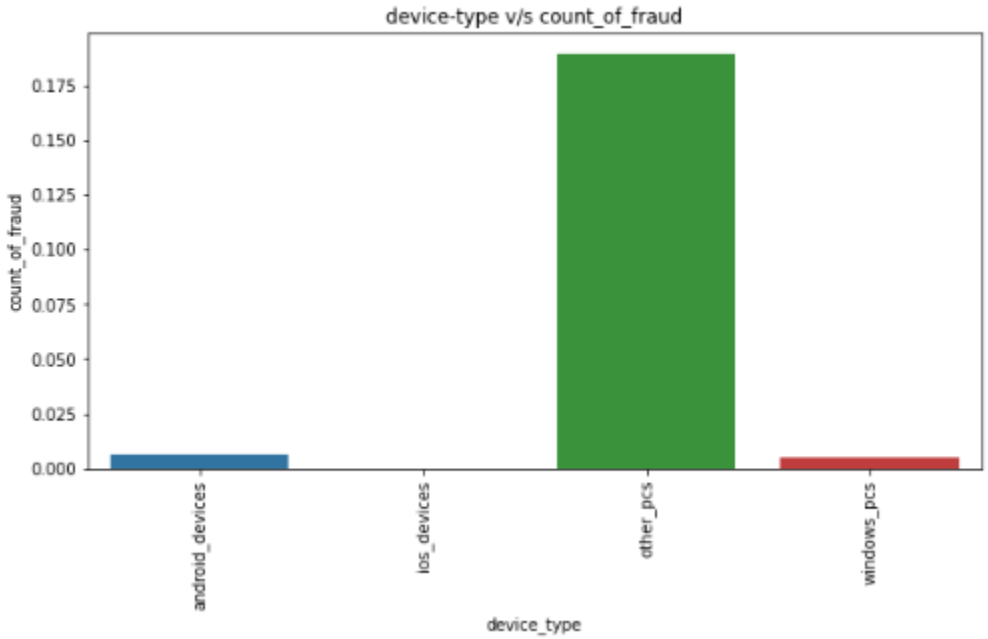


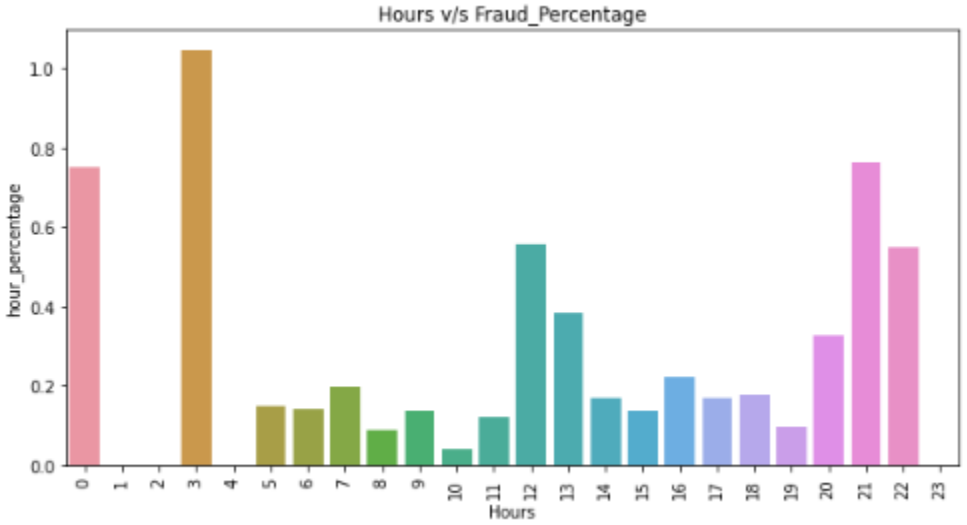


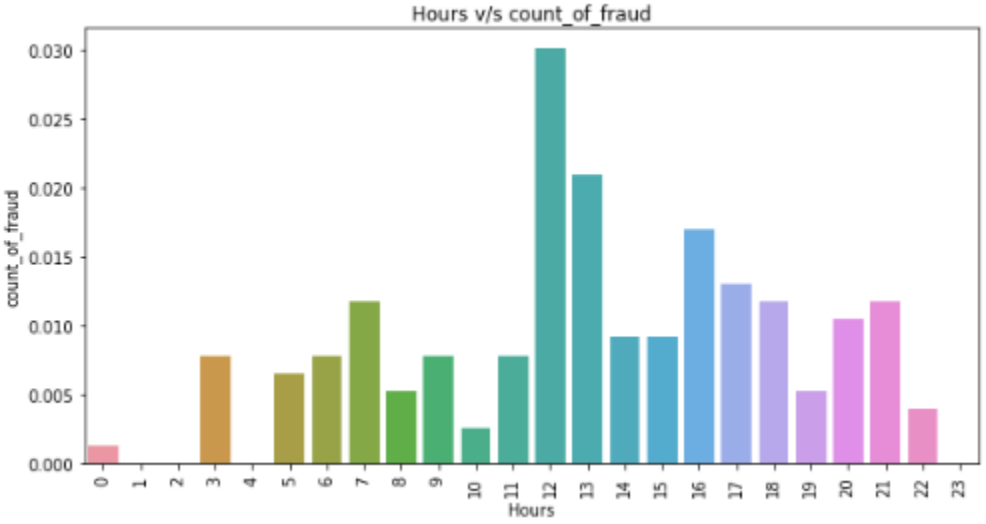

payment_method v/s Fraud_Percentage
payment_percentage
2.0
1.5
1.0
0.5
0.0
e_wallet_payments
other_debit_cards
sbi_atm_cum_debit_card
unified_payments_interface
visa_master_credit_cards
visa_master_debit_cards
payment_method

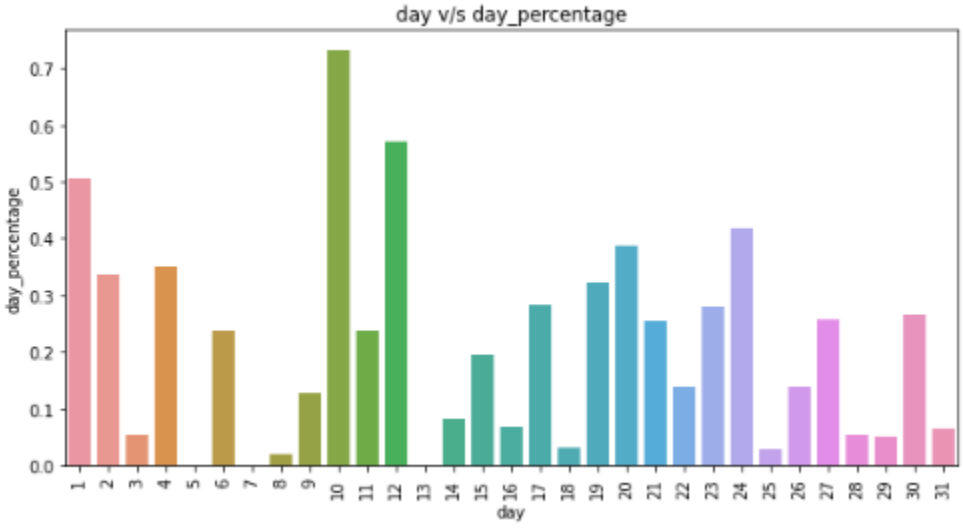

payment_method v/s Fraud_Percentage
count_of_fraud
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
e_wallet_payments
other_debit_cards
sbi_atm_cum_debit_card
unified_payments_interface
visa_master_credit_cards
visa_master_debit_cards
payment_method

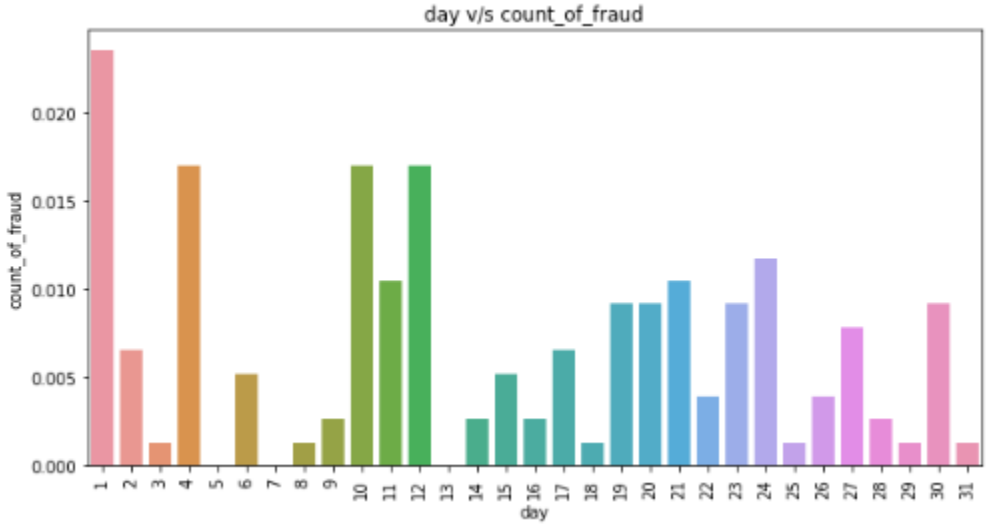

device_type v/s Fraud_Percentage
device_percentage
0.0
0.2
0.4
0.6
0.8
android_devices
ios_devices
other_pcs
windows_pcs
device_type

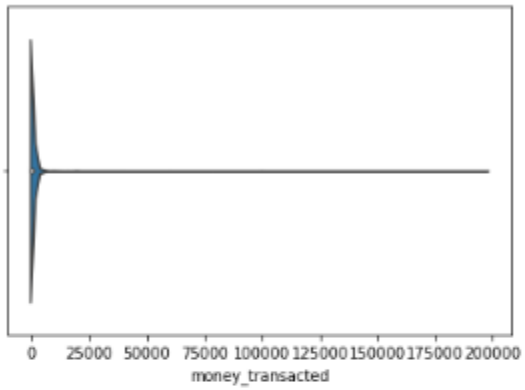

device-type v/s count_of_fraud
count_of_fraud
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
android_devices
ios_devices
other_pcs
windows_pcs
device_type

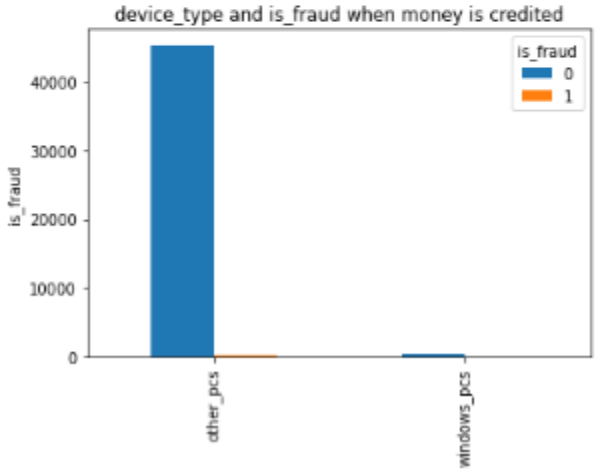

Hours v/s Fraud_Percentage
hour_percentage
Hours


Hours v/s count_of_fraud
count_of_fraud
Hours


day v/s day_percentage
day_percentage
day

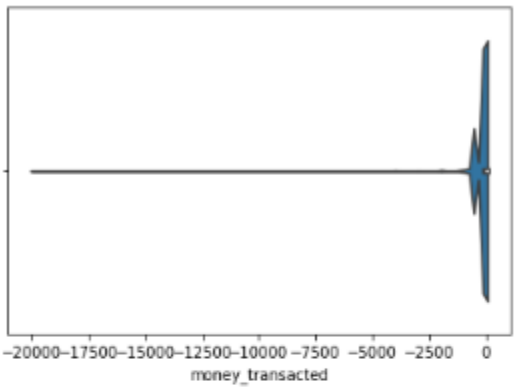

day v/s count_of_fraud
count_of_fraud
0.020
0.015
0.010
0.005
0.000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
day

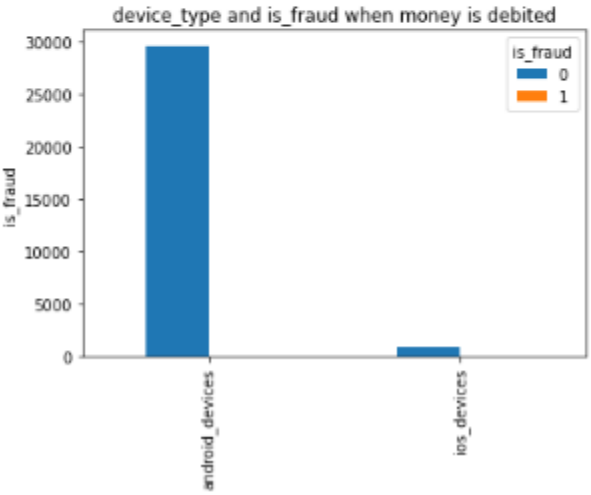

0 25000 50000 75000 100000 125000 150000 175000 200000
money_transacted


device_type and is_fraud when money is credited
is_fraud
0
1
40000
30000
20000
10000
0
is_fraud
other_pcs
windows_pcs


partner_pricing_category and is_fraud when money is credited
is_fraud
0
1
35000
30000
25000
20000
15000
10000
5000
0
is_fraud
0
1
2
4
partner_pricing_category


partner_category and is_fraud when money is credited
is_fraud
0
1
30000
25000
20000
15000
10000
5000
0
is_fraud
cat_1
cat_2
cat_3
cat_4
cat_5
cat_6
cat_7
cat_8
cat_9
partner_category


−20000 −17500 −15000 −12500 −10000 −7500 −5000 −2500 0
money_transacted


device_type and is_fraud when money is debited
is_fraud
0
1
30000
25000
20000
15000
10000
5000
0
is_fraud
android_devices
ios_devices


partner_pricing_category and is_fraud when money is debited
is_fraud
0
1
25000
20000
15000
10000
5000
0
is_fraud
0
2
4
partner_pricing_category


partner_category and is_fraud when money is debited
is_fraud
0
1
25000
20000
15000
10000
5000
0
is_fraud
cat_1
cat_2
cat_3
cat_4
cat_5
cat_6
partner_category

Conclusion/ Results

From visualization:

- Fraud has happened in high transactions amount
- Category 8> Category 3> Category 1 partner category had higher fraud percentage amount wise however count wise Category 1 > Category 2> Category 3.
- Visa master credit card> visa master debit card> other debit cards payment method had higher fraud percentage amount wise and count wise. Therefore, BI Gateway should increase scrutiny on these payment methods.
- Windows Pc and other Pcs has more fraud percentage both amount wise and count wise.
- Night time fraud percentage is higher, highest at 3 am amount wise however, count wise, mid-day time 11-16 is highest. Therefore , the higher amount frauds are happening in night time but max frauds are happening in day.

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.