Submitted by-Mallika Srivastava

Datamining Project-Spring 2021

Fraud Detection In Mobile Money Transactions

I. Aim

The project aims to use machine learning and data mining techniques to construct and validate a model to detect fraudulent mobile money transactions in financial domain.
We will use dataset named- 'Synthetic Financial Datasets for Fraud Detection' from Kaggle website (refer section II) for this purpose and create the model using python programming language and Google Colab notebook.

II. Introduction

For over last 10 years, money transactions via mobile/celluar platform are increasingly becoming popular among masses. It acts as a convenient tool for trades between customers and merchants by converting cash into 'electronic money'. This makes it easy to provide variety of financial services even to remote locations. Their increasing popularity can be inferred from the GSMA data which states in 2019, there were close to $2 billion in daily transactions via mobile money platforms.
While we are witnessing rapid development in technologies and emergence of new communication channels to further enhance the mobile money services for both businesses and common people, this platform has also become the target of ever evolving financial fraud.

- In 2019, 93% of total mobile transactions in 20 countries were blocked as fraudulent according to a report on the state of malware and mobile ad fraud released by Upstream[1].
- Even some of the well know secure platforms were made target for big frauds. For e.g.- Between 2015 and 2016, a fraudulent group conducted more than $1.5 million on fraudulent purchases via Apple Pay[2].

Therefore, this rise in fraudulent incidents has become one of the major business challenges of today's times.
While many studies and research on machine learning algorithms are happening in this field, considering the private nature of the financial transactions, it is difficult to directly work on the real data. However, synthetic data which matches closely to the real data can be used for this purpose.

III. Approach

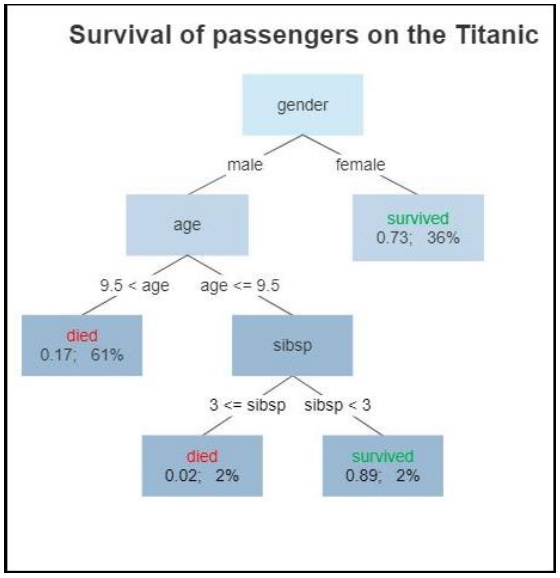
Find below our approach which includes dataset and methods used for this project-

a. **Dataset used in the project**- The dataset used is from the Kaggle website (pls find the link below). It is synthetic dataset generated using the simulator called PaySim as an approach to solve the problem lack of publicly available dataset. PaySim uses aggregated data from the private dataset to generate a synthetic dataset that resembles the normal operation of transactions and injects malicious behavior to later evaluate the performance of fraud detection methods[3].
Link to the data- **https://www.kaggle.com/ntnu-testimon/paysim1**

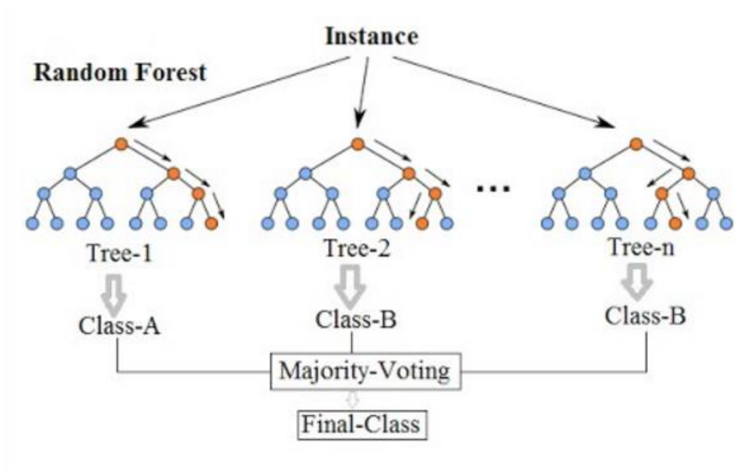
b. **Method Used-** Both supervised and un-supervised machine learning algorithms were used to conduct the analysis. For supervised learning, Decision Tree, Random Forest and Neural networks techniques were applied. For Un-supervised machine learning, Isolation Forest, SVM-one class and Local Outlier factor were used. Find below a brief description of each of the techniques-

i. Supervised learning techniques- infers a function from labeled training data

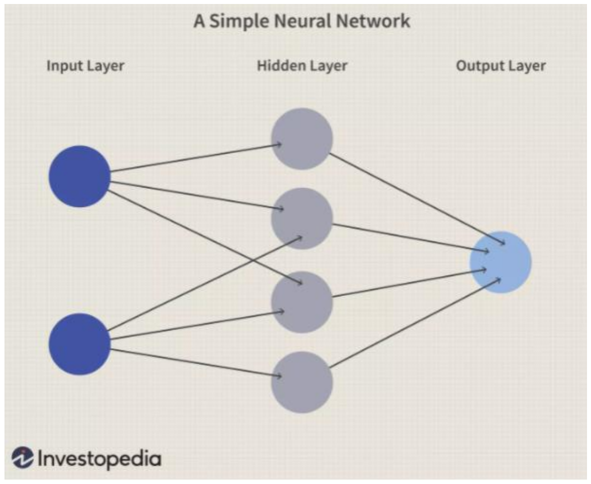
- **Decision Tree**- Decision Tree is based on the classification algorithm using tree-like model of decisions. It is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.
  Find below a sample example of the Decision tree[4]-



- **Random Forest**- Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set and therefore generally outperform decision trees performance. Find below a sample representation of the random forest[5]-

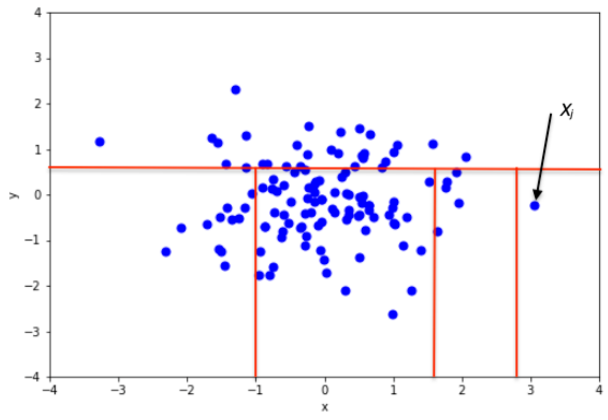


- **Neural Networks**- A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear. In a multi-layered perceptron (MLP), perceptron's are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map. Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis[6]**.**

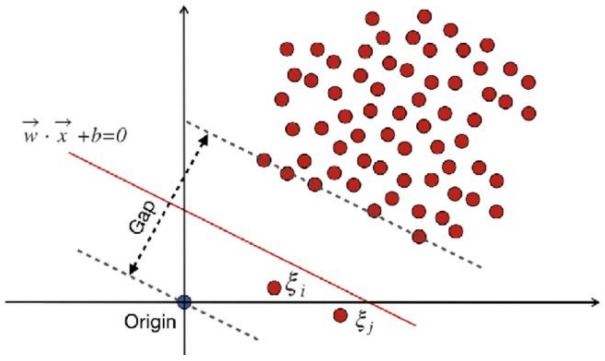


ii. Outlier detection through Un-supervised learning techniques: learns patterns from unlabeled data

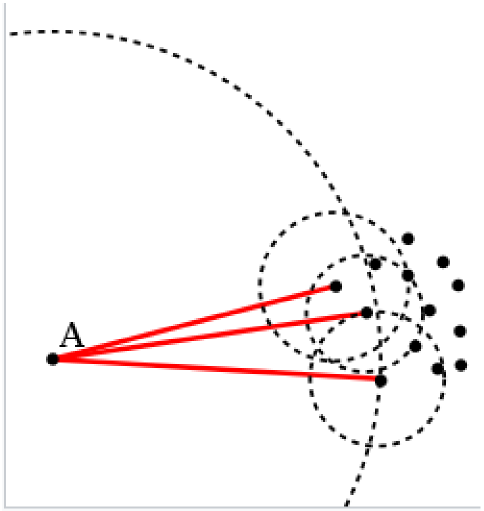
- **Isolation Forest**- Isolation forest is an unsupervised learning algorithm for anomaly detection that works on the principle of isolating anomalies instead of the most common techniques of profiling normal points. Since anomalies are "few and different", they are easier to “isolate” compared to normal points. Isolation Forest builds an ensemble of “Isolation Trees” (iTrees) for the data set, and anomalies are the points that have shorter average path lengths on the iTrees. The main advantage of this approach is it create a very fast algorithm with a low memory demand. An example of isolating an anomalous point x in a 2D Gaussian distribution[7].



- **SVM-one class**- A subfield of machine learning focused on identifying outliers in data is referred to as one-class classification. One-Class Classification, or OCC for short, involves fitting a model on the “normal” data and predicting whether new data is normal or an outlier/anomaly. A one-class classifier is fit on a training dataset that only has examples from the normal class. Once prepared, the model is used to classify new examples as either normal or not-normal, i.e. outliers or anomalies. The support vector machine, or SVM, algorithm developed initially for binary classification has been modified for one-class classification. When modeling one class, the SVM algorithm captures the density of the majority class and classifies examples on the extremes of the density function as outliers. This modification of SVM is referred to as One-Class SVM. Below is sample example of one class SVM where ξ are detected as outliers[8].



- **Local Outlier factor**- Local Outlier factor uses the approach of identifying outliers is to locate those examples that are far from the other examples in the feature space. It is a technique that attempts to harness the idea of nearest neighbors for outlier detection. Each example is assigned a scoring of how isolated or how likely it is to be outliers based on the size of its local neighborhood. Those examples with the largest score are more likely to be outliers. Below is a sample example[9]-



Basic idea of LOF: comparing the local density of a point with the densities of its neighbors. A has a much lower density than its neighbors.

IV. Methodology

The project was completed with the use of python language and its relevant libraries in the Google Colab notebook. The analysis was done in the following main steps-

a. Load and read the data
b. Data pre-processing
c. Exploratory data analysis
d. Feature selection and scaling
e. Running the algorithms and checking the accuracy

a. **Load and read the Data-** The dataset consists of 6362620 data points and 11 columns. Below are the details of the given 11 data attributes-

1. Step (Integer)- Each step is an hour of time in real world. The largest number for step is 744 (the 30th day).
2. Type (Categorical variable) – details the 5 Transaction types (CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER)
3. Amount (continuous variable)- Transaction amount in local currency
4. nameOrig(categorical variable)- who started the transaction
5. oldbalanceOrig (continuous variable) - Initial balance of sender before the transaction
6. newbalanceOrig (continuous variable) - The new balance of sender after the transaction
7. nameDest (categorical variable)- who received the transaction
8. oldbalanceDest (continuous variable) - Initial balance of receiver before the transaction
9. newbalanceDest (continuous variable) - New balance of receiver after the transaction
10. isFraud (categorical variable/target variable)- The status of a transaction (0 as legitimate and 1 as fraudulent)
11. isFlaggedFraud- The status that the system identified for a transaction.

b. Data Pre-processing- The 'isFlaggedFraud' column is removed as it is a system generated data and provides the same information as 'isfraud' column. We further derive following attributes from the given data-

- Error in origin account balance amount where
  Error = (New Balance + Transaction Amount) - Old Balance
  ```
  df['errorBalOrig'] = df['newbalanceOrig'] + df['amount'] - df['oldba
  lanceOrg']
  ```
- Error in destination account balance amount where
  #Error = (Old Balance + Transaction Amount) - New Balance
  ```
  df['errorBalDest'] = df['oldbalanceDest'] + df['amount'] - df['newba
  lanceDest']
  ```
- Find zero balance in origin account

```
df['zeroBalOrig'] = df['oldbalanceOrg','newbalanceOrig’]]
df['zeroBalOrig'] = df['zeroBalOrig'].apply(lambda x:1if x==0 else 0)
```

- Find zero balance in destination account

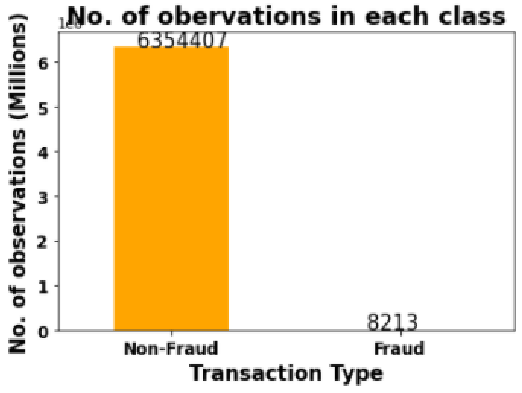
```
df['zeroBalDest'] = df[['oldbalanceDest','newbalanceDest']]
df['zeroBalDest'] = df['zeroBalDest'].apply(lambda x:1 if x== 0 else 0)
```

- Categorizing stepvalue into peak,mid ,least hour based on the proportion of fraud happening

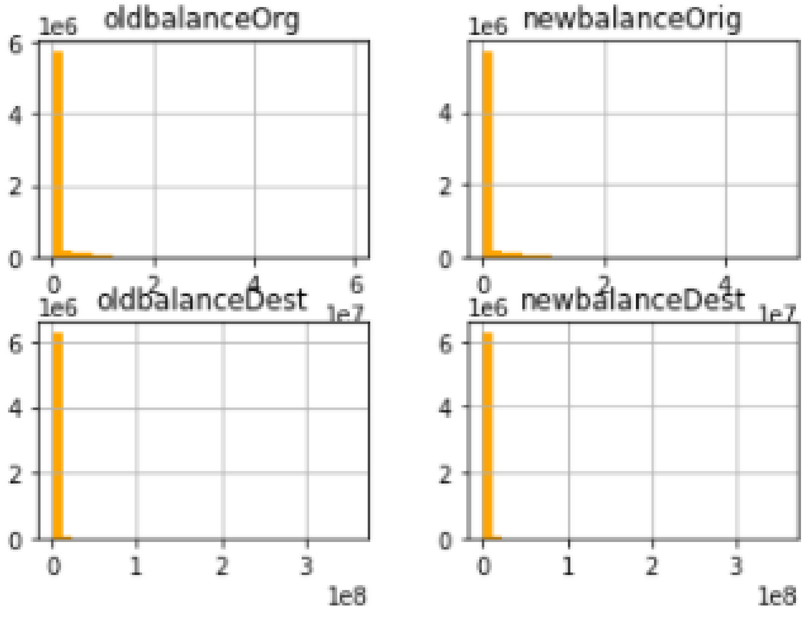
```
df['trans_period'] = df['trans_hour'].apply(lambda x: 'peak' if x in
 [2,3,4,5,6] else('mid' if x in [22,23,0,1,7,8] else 'safe'))
```

c. **Exploratory Data Analysis-**

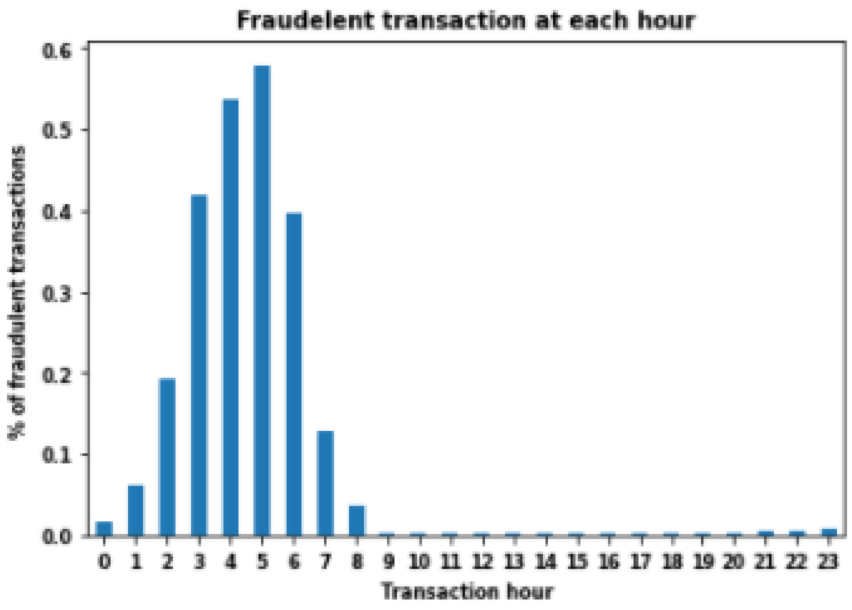
- The number of data points for fraud and non-fraud is imbalanced and only 0.13 % of data points are with value ‘fraud’



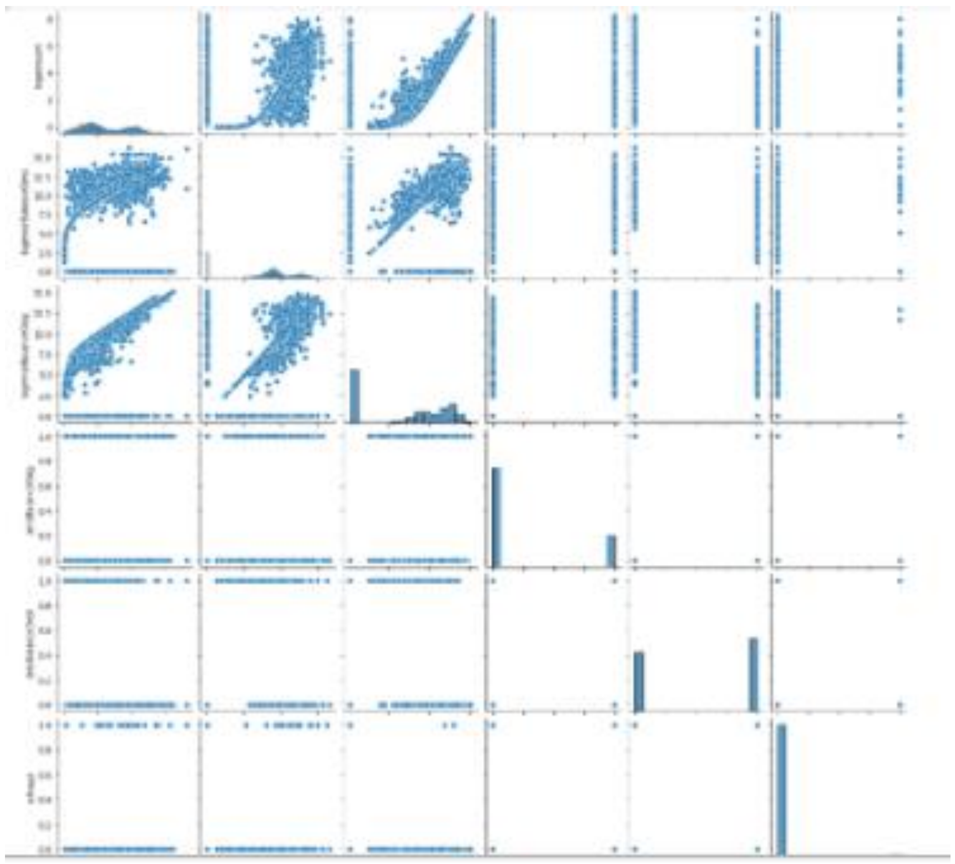
- The continuous variables are skewed to the right-



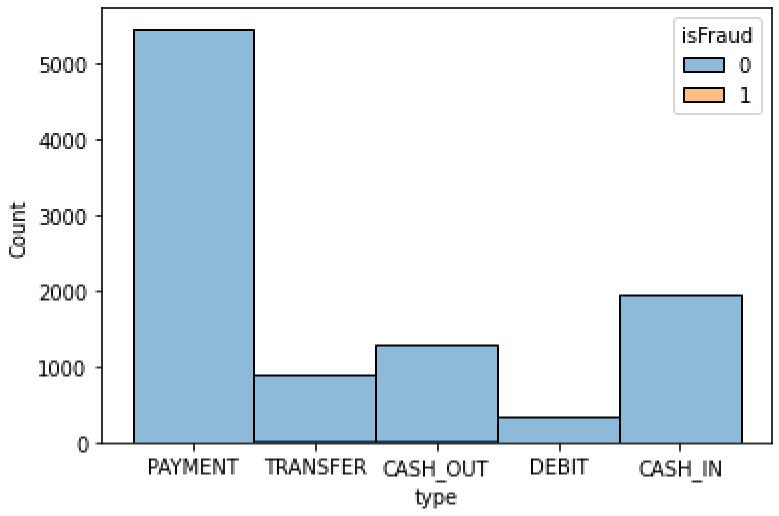
- Fraudulent transactions are at peak during late night hours-



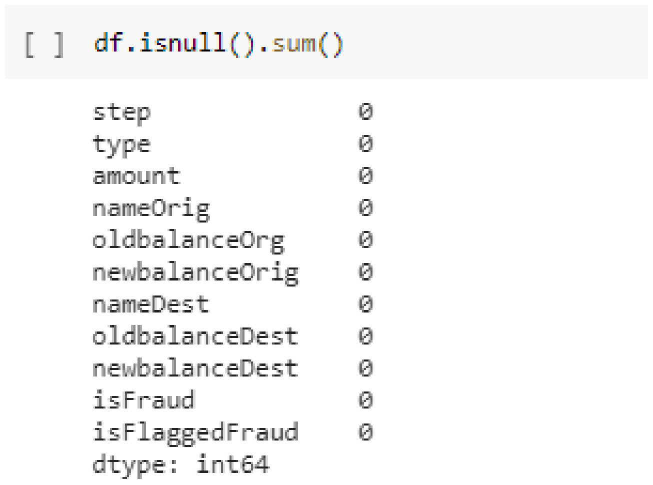
- Pair-plot indicates ‘income’ ,’balanceoriginal’ and ‘balancedetsination’ are correlated with ‘isFraud’ variable.



- Variable transaction ‘type’ doesn’t show much variance between ‘Fraud’ and ‘notfraud’. Therefore, fraud has no necessary preferred transactions type



- No missing values in data



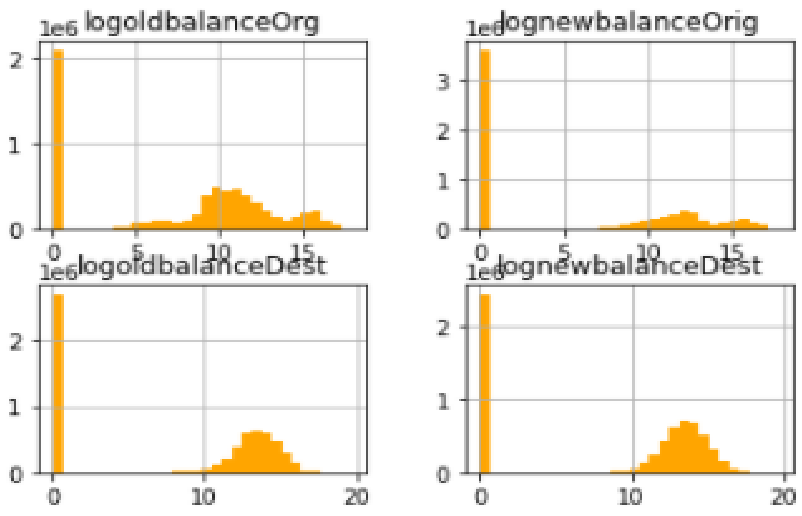
```
[ ]  df.isnull().sum()
```

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

d. Feature Selection and Scaling-

- All continuous variables were log transformed to correct for skewness. We can see normal distribution pattern post corrections, however there are still outliers which we can represent fraudulent transactions.

```
#log transforming all variables-
df["logamount"] = np.log1p(df["th_amount"])
df["logoldbalanceOrg"] = np.log1p(df["oldbalanceOrg"])
df["lognewbalanceOrig"] = np.log1p(df["newbalanceOrig"])
df["logoldbalanceDest"] = np.log1p(df["oldbalanceDest"])
df["lognewbalanceDest"] = np.log1p(df["newbalanceDest"])
df["logerrorBalanceDest"] = np.log1p(df["errorBalanceDest"])
df["logerrorBalanceOrig"] = np.log1p(df["errorBalanceOrig"])
```



- We dropped all original columns which has been log-transformed. Final data to be used to train the algorithm-

```
# Preparing new data with relevant columns
      df_new=df[['type','transactionPeriod','logamount','logerrorBal
      anceDest','logerrorBalanceOrig','zeroBalanceOrig','zeroBalance
      Dest','isFraud']]
```

- **Feature scaling**
  i. Continuous variables-using the standardization technique and StandardScaler()

```
#applying standard scaling
df_prep=df_new.copy()
col_names = ['logamount', 'logerrorBalanceDest', 'logerrorBala
nceOrig']
features = df_prep[col_names]
scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)
df_prep[col_names]=features
```

  ii. Categorical variables- By one hot encoding

```
df_new = pd.get_dummies(df_new,columns=['type','transactionPer
iod'])
```

e. Fitting the model

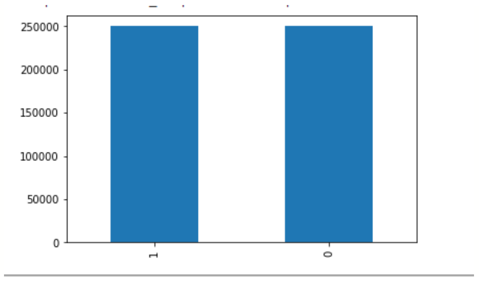
- Split into train and test data using stratify option to ensure equal distribution of both 'fraud' and 'non-fraud' cases

```
X_train,X_test,y_train,y_test=train_test_split(df_prep.drop("isFraud
",axis=1),df_new["isFraud"],test_size=0.1,random_state=1234,stratify
 = df_prep["isFraud"])
```

- **Correcting imbalanced dataset by down-sampling majority class and up-sampling minority class using skitlearn-imblearn package and SMOTE function.**

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=33)
X_train, y_train = smote.fit_sample(X_train, y_train)
```

```
1    250000
0    250000
Name: isFraud, dtype: int64
```



- Fitting the data into the algorithms-
  i. Decision Tree classifier

```
dt = DecisionTreeClassifier(random_state=1)
m1_dt=dt.fit(X_train,y_train)
```

  ii. Random Forest classifier

```
rfc=RandomForestClassifier(random_state=100)
m2_rfc=rfc.fit(X_train,y_train)
```

  iii. Neural Networks

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 32)                512       
_________________________________________________________________
dense_1 (Dense)              (None, 64)                2112      
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 520       
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 18        
=================================================================
Total params: 3,162
Trainable params: 3,162
Non-trainable params: 0
```

  iv. SVM One class

```
svm = OneClassSVM(kernel='rbf', nu=outlier_prop, gamma=0.0001)
svm.fit(train_normal)
```

  v. Isolation Forest

```
clf=IsolationForest(n_estimators=100, max_samples=len(train_no
rmal),contamination=outlier_prop,random_state=123, verbose=0)
clf.fit(train_normal)
```

vi. Local Outlier factor

```
lof=LocalOutlierFactor(n_neighbors=20, algorithm='auto',leaf_s
ize=30, metric='minkowski',p=2, metric_params=None, novelty=Tr
ue,contamination=outlier_prop)
lof.fit(train_normal)
```

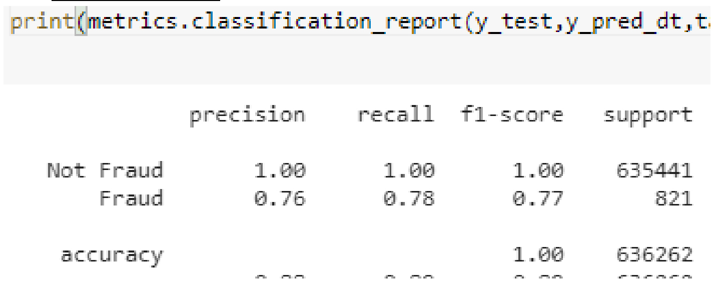
V. <u>Results- Interpretation and comparison</u>

a. Key metrics used to interpret and compare the results of the algorithms used-

- Confusion matrix-also known as an error matrix allows visualization of the performance of an algorithm wherein its each row represents the count of the predicted class and each column represents the instances in an actual class.
- Classification Report- consists of following components
  - recall- "how many of a class over the whole number of elements of the class"
  - precision- "how many are correctly classified among that class"
  - f1-score- harmonic mean between precision & recall
  - support- number of occurrences of the given class in the dataset
- ROC Curve-An ROC curve (or receiver operating characteristic curve) summarizes the performance of a binary classification model on the positive class. The x-axis indicates the False Positive Rate and the y-axis indicates the True Positive Rate. A good classifier will be far from line bisecting the square. A quantitative way to compare classifiers, is to measure the area under the curve (AUC)

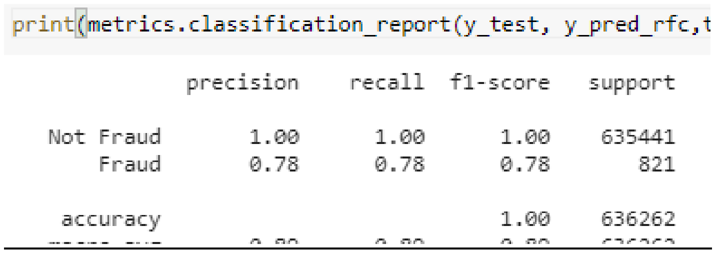
b. **Results for supervised learning algorithm**- We did not see much difference in the accuracy of the three algorithms used. The ROC curve is also same for all the three-

<u>Decision tree:</u>



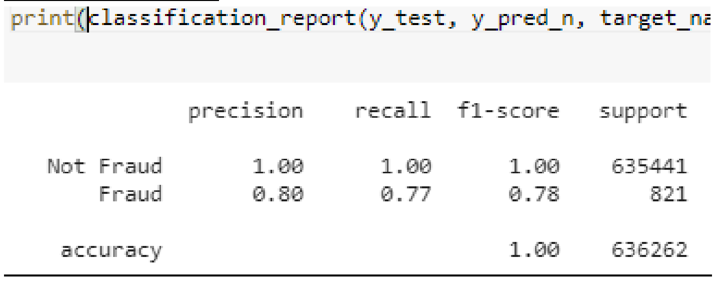
```
print(metrics.classification_report(y_test,y_pred_dt,t

              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00    635441
       Fraud       0.76      0.78      0.77       821

    accuracy                           1.00    636262
```

<u>Random Forest</u>

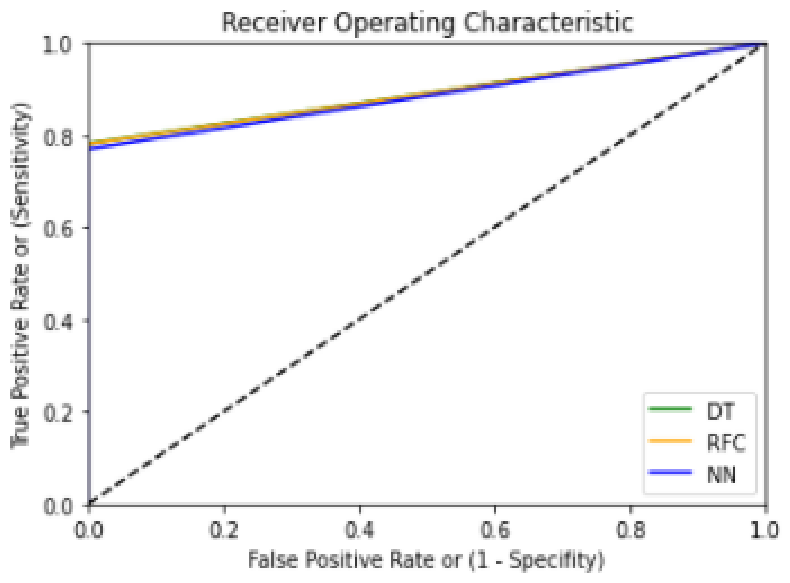


```
print(metrics.classification_report(y_test, y_pred_rfc,t

              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00    635441
       Fraud       0.78      0.78      0.78       821

    accuracy                           1.00    636262
```

Neural Networks:

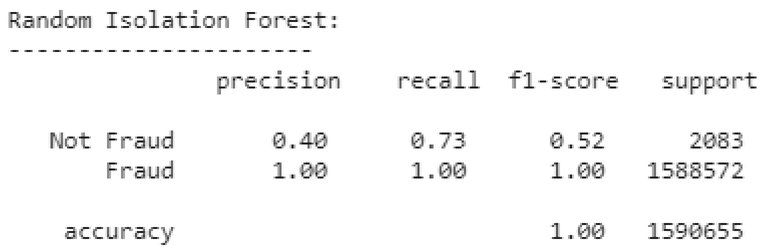


```
print(classification_report(y_test, y_pred_n, target_na

              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00    635441
       Fraud       0.80      0.77      0.78       821

    accuracy                           1.00    636262
```



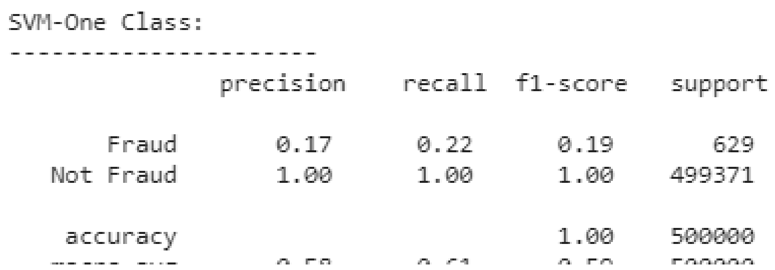
c. **Results for Un-supervised anomaly detection algorithms**- Isolation Forest performs best among all the algorithms. The ROC curve shows the same-

Isolation Forest



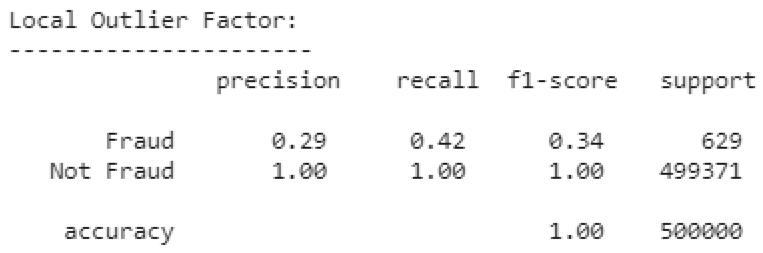
```
Random Isolation Forest:
----------------------
              precision    recall  f1-score   support

   Not Fraud       0.40      0.73      0.52      2083
       Fraud       1.00      1.00      1.00   1588572

    accuracy                           1.00   1590655
```

SVM-One class

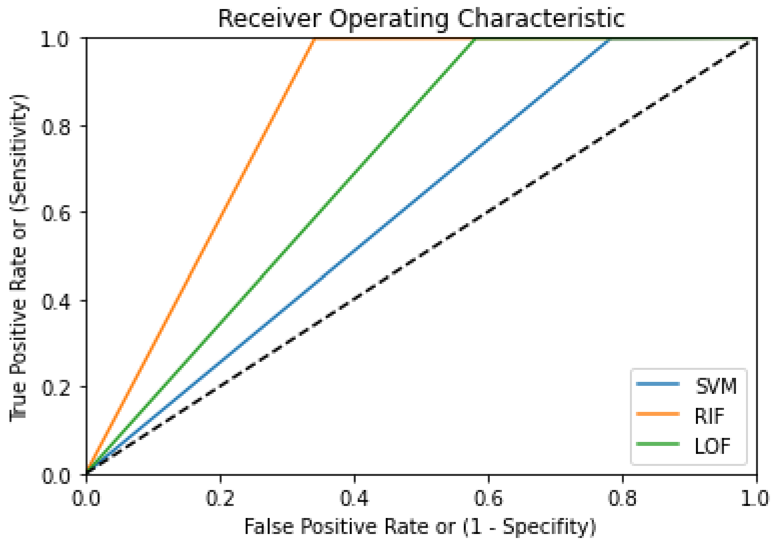


```
SVM-One Class:
----------------------
              precision    recall  f1-score   support

       Fraud       0.17      0.22      0.19       629
   Not Fraud       1.00      1.00      1.00    499371

    accuracy                           1.00    500000
```

Local Outlier Factor



```
Local Outlier Factor:
----------------------
              precision    recall  f1-score   support

       Fraud       0.29      0.42      0.34       629
   Not Fraud       1.00      1.00      1.00    499371

    accuracy                           1.00    500000
```



VI. Conclusion and Future Work

We conclude with some of notable findings of our analysis and challenges we still need deal with-

a. Findings-

- **Un-supervised methods of Outlier detection (notably Isolation Forest) fit better**- This shows most of the 'fraud' transactions are one-of-kinds and thus can be considered as an outlier. This also proves that supervised methods which assumes that both the 'fraud' and 'not fraud' classes have a pattern and tries to learns that pattern will not be very effective.
- **Late night transactions should have slightly high weight towards 'fraud'-** The analysis shows a trend of more fraud transactions happening towards the late-night hours. This also co-relates with our real-world experience as when people are sleeping, its easier for a fraud transaction to not get noticed immediately, thus making the transactions successful.

b. **Challenge to find higher recall and precision values together-** Since fraud in financial transactions can lead to high losses and also reputational loss to the platform on which transactions was done, therefore it is very important that we need to train our model to achieve both high precision and high recall values. We need high precision to ensure we don't block right transactions, as it will lead to financial loss and high recall to ensure we cover each fraudulent transaction as it will lead to financial/ reputational loss. This can happen only when we can have a very efficient model that can detect maximum numbers of fraud transactions accurately. To counter this challenge, we can try advanced outlier detections models and wherever possible real-life data to match the pattern more closely.

References-

1. Introduction to data mining by tan, steinbach, kumar solutions

2. Millions Are Being Lost To Apple Pay Fraud—Will Apple Card Come To The Rescue? (forbes.com)

3. **https://www.kaggle.com/ntnu-testimon/paysim1**

4. Decision tree learning - Wikipedia

5. Random forest - Wikipedia

6.https://www.investopedia.com/terms/n/neuralnetwork.asp#:~:text=Neural%20networks%20are%20a%20series,fraud%20detection%20and%20risk%20assessment

7. https://en.wikipedia.org/wiki/Isolation_forest

8. https://machinelearningmastery.com/one-class-classification-algorithms/; https://www.researchgate.net/figure/Overview-of-one-class-Support-Vector-Machine-SVM_fig3_317711827

9. https://en.wikipedia.org/wiki/Local_outlier_factor

10. Fraud Detection in Mobile Money Transactions Using Machine Learning (iastate.edu)