DETECTION OF FRAUD IN FINANCIAL TRANSACTIONS

Submitted By

RAKTIM MUKHOPADHYAY
ROLL NO: GCECTB-R15-3019
REGN. NO.: 151130110019 OF 2015-16

Under the guidance of

PROF. ARNAB CHAKRABORTY

B.Sc. (Hons), B. Tech & MTech (CS), MBA (HR & MM),
Six Sigma Master Black Belt certified by IQF (USA).
AMT by AIMA, NLP and PMP Trained.
ITIL Foundation certified by APMG (UK).
Control & Automation System certified by ISA (USA).

In partial fulfillment for the award of the certificate of
MACHINE LEARNING WITH PYTHON

Arnab Chakraborty

Globsyn Skills Development (P) Ltd
Kolkata

Document sign date :22 Jul, 2018

CONTENTS

Arnab Chakraborty

Globsyn Skills Development (P) Ltd ★ Kolkata

Document sign date :22 Jul, 2018

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my faculty Prof. Arnab Chakraborty for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him/her time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

With gratitude,

RAKTIM MUKHOPADHYAY

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. ★ Kolkata

Document sign date :22 Jul, 2018

OBJECTIVE

The aim of the project is to compare properties of machine learning algorithms to learn and apply learned knowledge in the task of prediction. The type of learning is limited to Supervised learning. The algorithms which will be applied are Logistic Regression, K - nearest neighbors, Naïve-Bayes and Decision tree.

In this project we develop a Fraud Detection Framework in Financial Payment Services over an imbalanced synthetic financial dataset generated by Paysim having over 6.5 million financial transactions with using Logistic Regression, Decision Tree, Naive Bayes, Random and KNN.

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

SCOPE

The broad scope of the **Detection of Fraud in Financial Transactions** project includes:

- The system will be available on an online banking system for 24x365 access to the Cyber Security Personnel of the bank.
- The system will support Machine Learning based detection of Fraud Transactions.
- We can predict fraud in a large volume of transactions by applying cognitive computing technologies to the raw data.
- This is the reason why machine learning algorithms will be used by banks for preventing fraud for their clients.

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

DATA DESCRIPTION

SOURCE OF DATA: https://www.kaggle.com/ntnu-testimon/paysim1

There is a lack of public available datasets on financial services and specially in the emerging mobile money transactions domain. Financial datasets are important to many researchers and in particular to us performing research in the domain of fraud detection. Part of the problem is the intrinsically private nature of financial transactions, that leads to no publicly available datasets.

We present a synthetic dataset generated using the simulator called **PaySim** as an approach to such a problem. 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.

- **step (int64)** - maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation).
- **type (object)** - CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.
- **Amount (float64)** - amount of the transaction in local currency.
- **nameOrig(object)** - customer who started the transaction.
- **oldbalanceOrg (float64)** - initial balance before the transaction.
- **newbalanceOrig (float64)** - new balance after the transaction.
- **nameDest (obj)**- customer who is the recipient of the transaction.
- **oldbalanceDest (float64)** - initial balance recipient before the transaction.
- **newbalanceDest (float64)** - new balance recipient after the transaction.
- **isFraud (int64)** - This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.
- **isFlaggedFraud (int64)** - The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

MODELS

- **NAÏVE BAYES**

The Naive Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

Likelihood Class Prior Probability

$$P(c\mid x)=\frac{P(x\mid c)P(c)}{P(x)}$$

Posterior Probability Predictor Prior Probability

$$P(c\mid X) = P(x_1\mid c)\times P(x_2\mid c)\times \cdots \times P(x_n\mid c)\times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

Arnab Chakraborty

Globsyn Skills Development (P) Ltd Kolkata

Document sign date :22 Jul, 2018

- **K-NEAREST NEIGHBOURS**

The KNN algorithm is a robust and versatile classifier that is often used as a benchmark for more complex classifiers such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM). Despite its simplicity, KNN can outperform more powerful classifiers and is used in a variety of applications such as economic forecasting, data compression and genetics. For example, KNN was leveraged in a 2006 study of functional genomics for the assignment of genes based on their expression profiles.

KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labelled dataset consisting of training observations (x,y) and would like to capture the relationship between x and y. More formally, our goal is to learn a function $h:X\rightarrow Y$ so that given an unseen observation x, h(x) can confidently predict the corresponding output y.

The KNN classifier is also a non-parametric and instance-based learning algorithm.

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given "unseen" observation. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance given by

$$\begin{aligned} d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) &= \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}. \end{aligned}$$

but other measures can be more suitable for a given setting and include the Manhattan, Chebyshev and Hamming distance.

Arnab Chakraborty

Globsyn Skills Development (P) Ltd ★ Kolkata

Document sign date :22 Jul, 2018

- **DECISION TREE**

Decision tree learning 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). It is one of the predictive modelling approaches used in statistics, data mining and machine learning. 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.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making). This page deals with decision trees in data mining.

- **LOGISTIC REGRESSION**

Logistic regression is named for the function used at the core of the method, the logistic function.
The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.
1 / (1 + e^-value)
Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform.

EXPLORATORY DATA ANALYSIS AND DATA CLEANING

Arnab Chakraborty

Globsyn Skills Development (P) Ltd * Kolkata

Document sign date :22 Jul, 2018

EXPLORATORY DATA ANALYSIS

1 EXPLORATORY DATA ANALYSIS

The provided data has the financial transaction data as well as the target variable isFraud, which is the actual fraud status of the transaction and isFlaggedFraud is the indicator which the simulation is used to flag the transaction using some threshold. The goal should be how we can improve and come up with better threshold to capture the fraud transaction.

```
In [ ]: # import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        import seaborn as sns
        import matplotlib.pyplot as plt
        import matplotlib.cm as cm
        %matplotlib inline
```

```
In [32]: # loading dataset
         try:
             FraudTransactions=pd.read_csv('C:/Users\Raktim\Desktop\Python\PS_20174392719_1491
         except:
             print('Database not able to load')
         df=FraudTransactions
```

Test if there any missing values in DataFrame. It turns out there are no obvious missing values but, as we will see below, this does not rule out proxies by a numerical value like 0.

```
In [5]: print(df.isnull().values.any())
```

```
False
```

Quickly look at the dataset sample and other properties.

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

```
In [6]: print(df.head())
        print(df.describe())
        print(df.info())
```

```
   step      type    amount     nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1   PAYMENT   9839.64  C1231006815       170136.0       160296.36
1     1   PAYMENT   1864.28  C1666544295        21249.0        19384.72
```

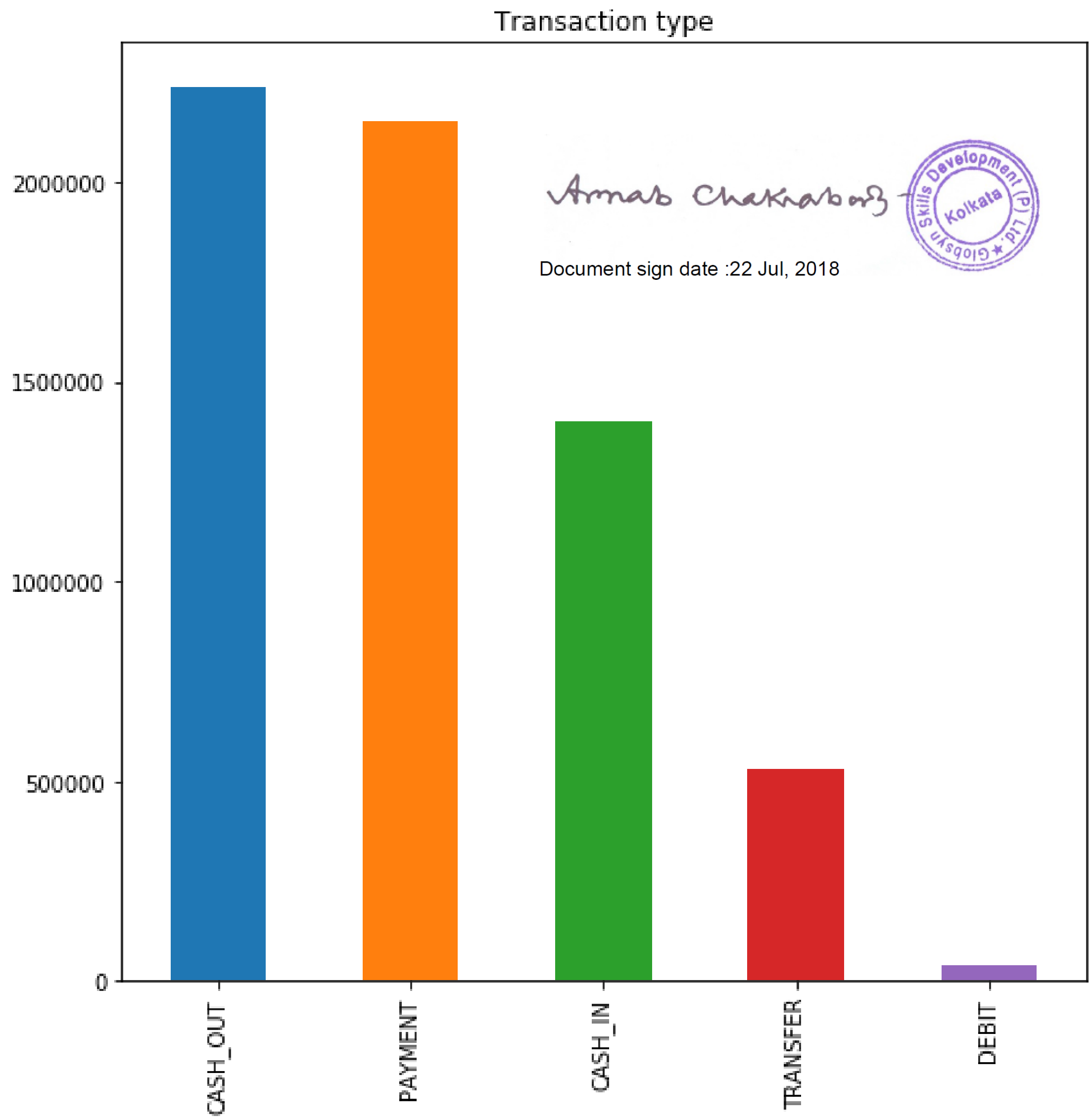
```
2      1  TRANSFER    181.00  C1305486145         181.0           0.00
3      1  CASH_OUT    181.00   C840083671         181.0           0.00
4      1   PAYMENT  11668.14  C2048537720       41554.0       29885.86

      nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155             0.0             0.0        0               0
1  M2044282225             0.0             0.0        0               0
2   C553264065             0.0             0.0        1               0
3    C38997010         21182.0             0.0        1               0
4  M1230701703             0.0             0.0        0               0
               step        amount  oldbalanceOrg  newbalanceOrig  \
count  6.362620e+06  6.362620e+06   6.362620e+06    6.362620e+06
mean   2.433972e+02  1.798619e+05   8.338831e+05    8.551137e+05
std    1.423320e+02  6.038582e+05   2.888243e+06    2.924049e+06
min    1.000000e+00  0.000000e+00   0.000000e+00    0.000000e+00
25%    1.560000e+02  1.338957e+04   0.000000e+00    0.000000e+00
50%    2.390000e+02  7.487194e+04   1.420800e+04    0.000000e+00
75%    3.350000e+02  2.087215e+05   1.073152e+05    1.442584e+05
max    7.430000e+02  9.244552e+07   5.958504e+07    4.958504e+07

       oldbalanceDest  newbalanceDest       isFraud  isFlaggedFraud
count    6.362620e+06    6.362620e+06  6.362620e+06    6.362620e+06
mean     1.100702e+06    1.224996e+06  1.290820e-03    2.514687e-06
std      3.399180e+06    3.674129e+06  3.590480e-02    1.585775e-03
min      0.000000e+00    0.000000e+00  0.000000e+00    0.000000e+00
25%      0.000000e+00    0.000000e+00  0.000000e+00    0.000000e+00
50%      1.327057e+05    2.146614e+05  0.000000e+00    0.000000e+00
75%      9.430367e+05    1.111909e+06  0.000000e+00    0.000000e+00
max      3.560159e+08    3.561793e+08  1.000000e+00    1.000000e+00
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
step              int64
type              object
amount            float64
nameOrig          object
oldbalanceOrg     float64
newbalanceOrig    float64
nameDest          object
oldbalanceDest    float64
newbalanceDest    float64
isFraud           int64
isFlaggedFraud    int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
None
```

Arnab Chakraborty

Document sign date :22 Jul, 2018

Globsyn Skills Development (P) Ltd Kolkata

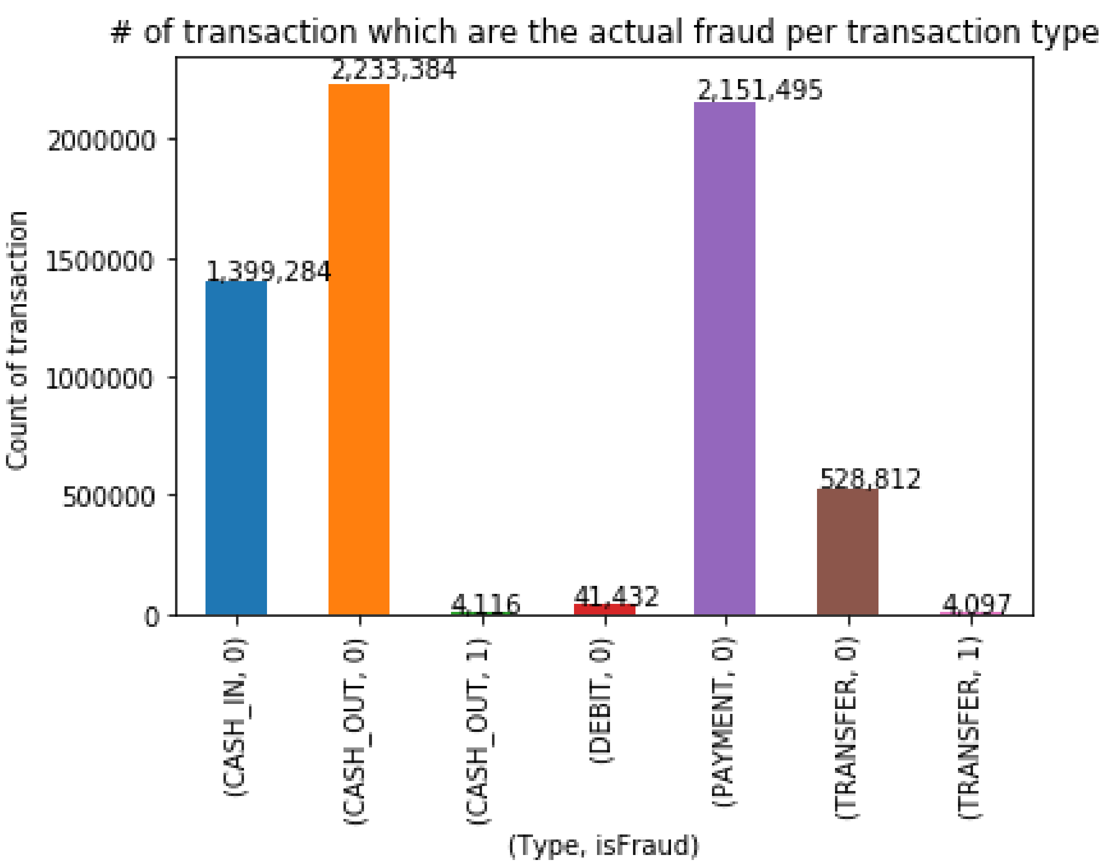
```
In [7]: f, ax = plt.subplots(1, 1, figsize=(8, 8))
        df.type.value_counts().plot(kind='bar', title="Transaction type", ax=ax, figsize=(8,8))
        plt.show()
```




```
In [8]: print('\n The types of fraudulent transactions are {}'.format(\
        list(df.loc[df.isFraud == 1].type.drop_duplicates().values))) # only 'CASH_OUT'
                                                                      # & 'TRANSFER'

        dfFraudTransfer = df.loc[(df.isFraud == 1) & (df.type == 'TRANSFER')]
        dfFraudCashout = df.loc[(df.isFraud == 1) & (df.type == 'CASH_OUT')]

        print ('\n The number of fraudulent TRANSFERs = {}'.\
```

```
        format(len(dfFraudTransfer))) # 4097

print ('\n The number of fraudulent CASH_OUTs = {}'.\
        format(len(dfFraudCashout))) # 4116

ax = df.groupby(['type', 'isFraud']).size().plot(kind='bar')
ax.set_title("# of transaction which are the actual fraud per transaction type")
ax.set_xlabel("(Type, isFraud)")
ax.set_ylabel("Count of transaction")
for p in ax.patches:
    ax.annotate(str(format(int(p.get_height()), ',d')), (p.get_x(), p.get_height()*1.0
```

```
The types of fraudulent transactions are ['TRANSFER', 'CASH_OUT']

The number of fraudulent TRANSFERs = 4097

The number of fraudulent CASH_OUTs = 4116
```



Arnab Chakraborty

Globsyn Skills Development (P) Ltd Kolkata

Document sign date :22 Jul, 2018

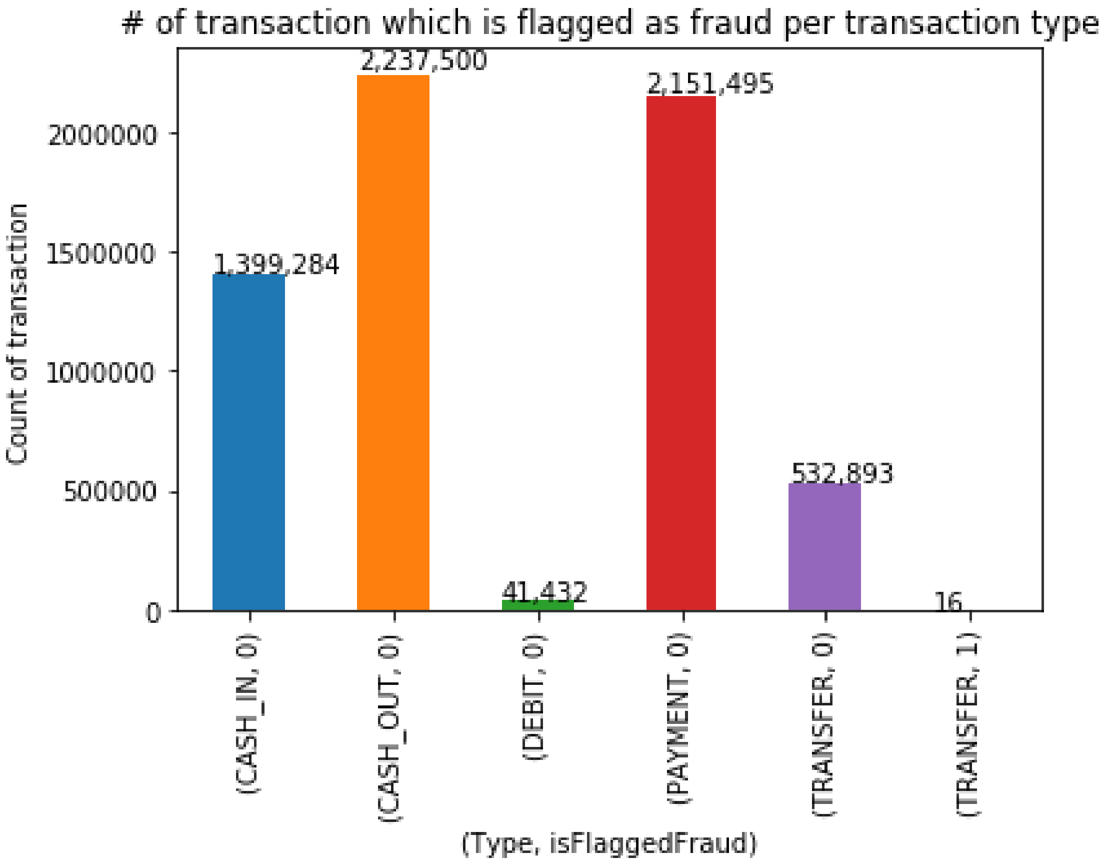
We find that of the five types of transactions, fraud occurs only in two of them 'TRANSFER' where money is sent to a customer / fraudster and 'CASH_OUT' where money is sent to a merchant who pays the customer / fraudster in cash. Remarkably, the number of fraudulent TRANSFERs almost equals the number of fraudulent CASH_OUTs. This gives us an insight into the modus operandi of fraudulent transactions in this dataset, namely, fraud is committed by first transferring out funds to another account which subsequently cashes it out.

There are 2 flags which stand out to me and it's interesting to look onto: isFraud and isFlaggedFraud column. From the hypothesis, isFraud is the indicator which indicates the actual fraud transactions whereas isFlaggedFraud is what the system prevents the transaction due to some thresholds being triggered. Let's quickly what kinds of transaction are being flagged and are fraud...

It turns out that the origin of isFlaggedFraud is unclear, contrasting with the description provided. The 16 entries (out of 6 million) where the isFlaggedFraud feature is set do not seem to correlate with any explanatory variable. The data is described as isFlaggedFraud being set when an attempt is made to 'TRANSFER' an 'amount' greater than 200,000. In fact, as shown below, isFlaggedFraud can remain not set despite this condition being met

```
In [9]: ax = df.groupby(['type', 'isFlaggedFraud']).size().plot(kind='bar')
        ax.set_title("# of transaction which is flagged as fraud per transaction type")
        ax.set_xlabel("(Type, isFlaggedFraud)")
        ax.set_ylabel("Count of transaction")
        for p in ax.patches:
            ax.annotate(str(format(int(p.get_height()), ',d')), (p.get_x(), p.get_height()*1.0

        print('\nThe type of transactions in which isFlaggedFraud is set: \
        {}'.format(list(df.loc[df.isFlaggedFraud == 1].type.drop_duplicates())))
                                                                      # only 'TRANSFER'

        dfTransfer = df.loc[df.type == 'TRANSFER']
        dfFlagged = df.loc[df.isFlaggedFraud == 1]
        dfNotFlagged = df.loc[df.isFlaggedFraud == 0]

        print('\nMin amount transacted when isFlaggedFraud is set= {}'\
              .format(dfFlagged.amount.min())) # 353874.22

        print('\nMax amount transacted in a TRANSFER where isFlaggedFraud is not set=\{}'
              .format(dfTransfer.loc[dfTransfer.isFlaggedFraud == 0].amount.max()))
```

```
The type of transactions in which isFlaggedFraud is set: ['TRANSFER']

Min amount transacted when isFlaggedFraud is set= 353874.22

Max amount transacted in a TRANSFER where isFlaggedFraud is not set=\92445516.64
```



Let's look at those records of the transfers where isFlaggedFraud is set and compare with the records which the system cannot catch'em. The plot below will also focus only on transfer transaction type.

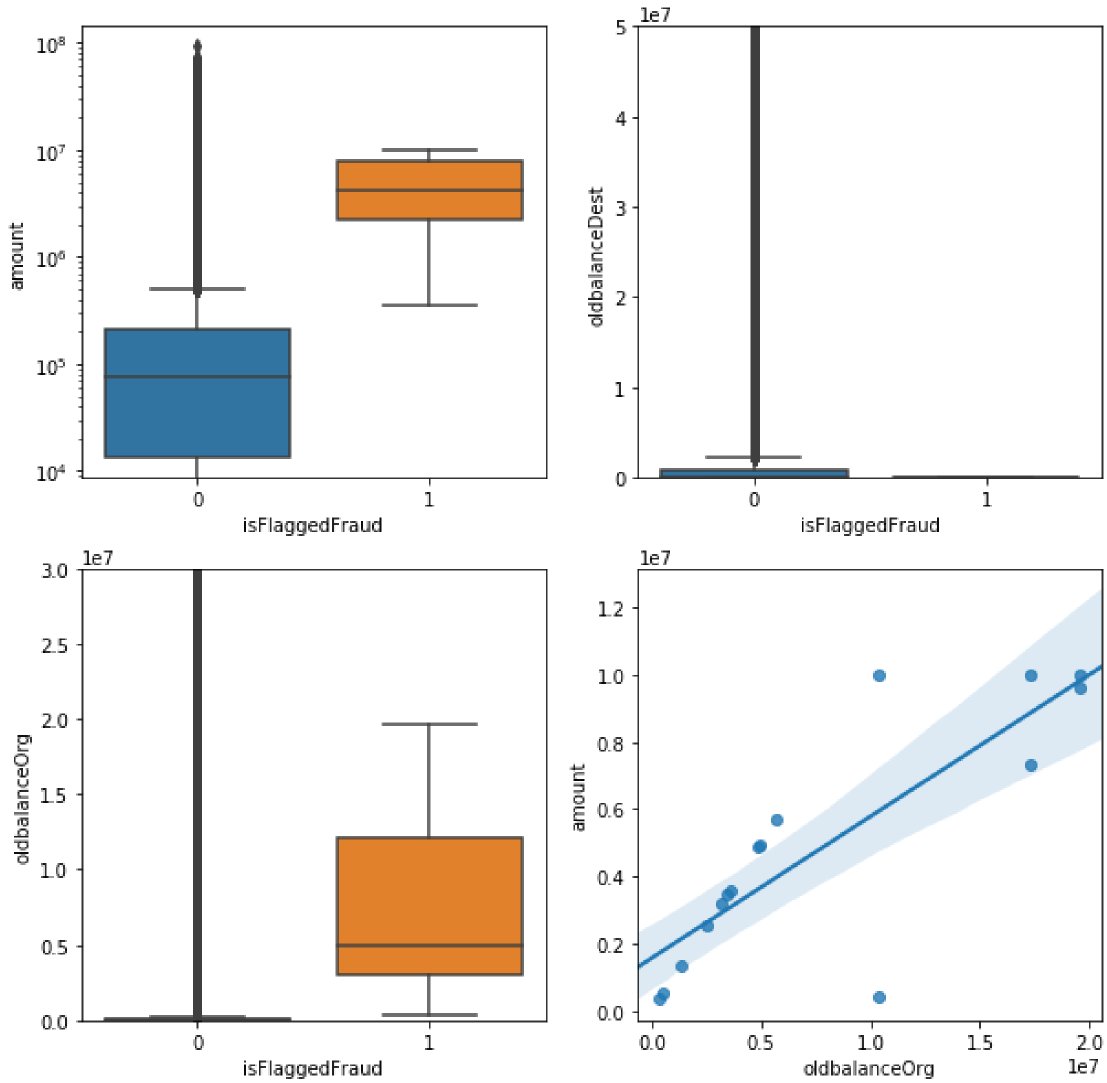
```
In [10]: fig, axs = plt.subplots(2, 2, figsize=(10, 10))
         tmp = df.loc[(df.type == 'TRANSFER'), :]

         a = sns.boxplot(x = 'isFlaggedFraud', y = 'amount', data = df, ax=axs[0][0])
         axs[0][0].set_yscale('log')
         b = sns.boxplot(x = 'isFlaggedFraud', y = 'oldbalanceDest', data = df, ax=axs[0][1])
         axs[0][1].set(ylim=(0, 0.5e8))
         c = sns.boxplot(x = 'isFlaggedFraud', y = 'oldbalanceOrg', data=df, ax=axs[1][0])
         axs[1][0].set(ylim=(0, 3e7))
         d = sns.regplot(x = 'oldbalanceOrg', y = 'amount', data=df.loc[(df.isFlaggedFraud ==1
         plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a no
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Arnab Chakraborty

Skills Development (P) Ltd. Kolkata Globsyn

Document sign date :22 Jul, 2018



By looking at the visualisations we think that isFlaggedFraud might depend on oldbalanceDest,which is 0 and some threshold on the amount variable. Futher studies might confirm our assumption

Can oldBalanceDest and newBalanceDest determine isFlaggedFraud being set? The old is identical to the new balance in the origin and destination accounts, for every TRANSFER where isFlaggedFraud is set. This is presumably because the transaction is halted. Interestingly, oldBalanceDest = 0 in every such transaction. However, as shown below, sinceisFlaggedFraud can remain not set in TRANSFERS where oldBalanceDest and newBalanceDest can both be 0, these conditions do not determine the state of isFlaggedFraud.

```
In [11]: print('\nThe number of TRANSFERs where isFlaggedFraud = 0, yet oldBalanceDest = 0 and\
          newBalanceDest = 0: {}'.\
         format(len(dfTransfer.loc[(dfTransfer.isFlaggedFraud == 0) & \
         (dfTransfer.old                                    balanceDest == 0)]))) # 4158
```

```
The number of TRANSFERs where isFlaggedFraud = 0, yet oldBalanceDest = 0 and newBalanceDest = (
```

isFlaggedFraud being set cannot be thresholded on oldBalanceOrig since the corresponding range of values overlaps with that for TRANSFERs where isFlaggedFraud is not set (see below). Note that we do not need to consider newBalanceOrigsince it is updated only after the transaction, whereas isFlaggedFraud would be set before the transaction takes place.

```
In [12]: print('\nMin, Max of oldbalanceOrg for isFlaggedFraud = 1 TRANSFERs: {}'.\
         format([round(dfFlagged.oldbalanceOrg.min()), round(dfFlagged.oldbalanceOrg.max())]))

         print('\nMin, Max of oldBalanceOrig for isFlaggedFraud = 0 TRANSFERs where \
         oldBalanceOrig = \
         newBalanceOrig: {}'.format(\
         [dfTransfer.loc[(dfTransfer.isFlaggedFraud == 0) & (dfTransfer.oldbalanceOrg \
         == dfTransfer.newbalanceOrig)].oldbalanceOrg.min(), \
         round(dfTransfer.loc[(dfTransfer.isFlaggedFraud == 0) & (dfTransfer.oldbalanceOrg \
                        == dfTransfer.newbalanceOrig)].oldbalanceOrg.max())]))
```

```
Min, Max of oldbalanceOrg for isFlaggedFraud = 1 TRANSFERs: [353874.0, 19585040.0]

Min, Max of oldBalanceOrig for isFlaggedFraud = 0 TRANSFERs where oldBalanceOrig = newBalanceO
```

Can isFlaggedFraud be set based on seeing a customer transacting more than once? Note that duplicate customer names don't exist within transactions where isFlaggedFraud is set, but duplicate customer names exist within transactions where isFlaggedFraud is not set. It turns out that originators of transactions that have isFlaggedFraud set have transacted only once. Very few destination accounts of transactions that have isFlaggedFraud set have transacted more than once.

```
In [13]: print('\nHave originators of transactions flagged as fraud transacted more than \
         once? {}'\
         .format((dfFlagged.nameOrig.isin(pd.concat([dfNotFlagged.nameOrig, \
                                            dfNotFlagged.nameDest]))).any())) # False

         print('\nHave destinations for transactions flagged as fraud initiated\
          other transactions? \
         {}'.format((dfFlagged.nameDest.isin(dfNotFlagged.nameOrig)).any())) # False

         # Since only 2 destination accounts of 16 that have 'isFlaggedFraud' set have been
         # destination accounts more than once,
         # clearly 'isFlaggedFraud' being set is independent of whether a
         # destination account has been used before or not

         print('\nHow many destination accounts of transactions flagged as fraud have been \
         destination acc
         .format(sum(dfF                                  .ameDest)))) # 2
```

```
Have originators of transactions flagged as fraud transacted more than once? False

Have destinations for transactions flagged as fraud initiated other transactions? False

How many destination accounts of transactions flagged as fraud have been destination accounts 
```

It can be easily seen that transactions with isFlaggedFraud set occur at all values of step, similar to the complementary set of transactions. Thus isFlaggedFraud does not correlate with step either and is therefore seemingly unrelated to any explanatory variable or feature in the data

Conclusion: Although isFraud is always set when isFlaggedFraud is set, since isFlaggedFraud is set just 16 times in a seemingly meaningless way, we can treat this feature as insignificant and discard it in the dataset without loosing information.

Are expected merchant accounts accordingly labelled? It was stated that CASH_IN involves being paid by a merchant (whose name is prefixed by 'M'). However, as shown below, the present data does not have merchants making CASH_IN transactions to customers.

```
In [14]: print('\nAre there any merchants among originator accounts for CASH_IN \
         transactions? {}'.format(\
         (df.loc[df.type == 'CASH_IN'].nameOrig.str.contains('M')).any()))
```

```
Are there any merchants among originator accounts for CASH_IN transactions? False
```

Similarly, it was stated that CASH_OUT involves paying a merchant. However, for CASH_OUT transactions there are no merchants among the destination accounts.

```
In [15]: print('\nAre there any merchants among destination accounts for CASH_OUT \
         transactions? {}'.format(\
         (df.loc[df.type == 'CASH_OUT'].nameDest.str.contains('M')).any()))
```

```
Are there any merchants among destination accounts for CASH_OUT transactions? False
```

In fact, there are no merchants among any originator accounts. Merchants are only present in destination accounts for all PAYMENTS

```
In [16]: print('\nAre there merchants among any originator accounts? {}'.format(\
               df.nameOrig.str.contains('M').any())) # False

         print('\nAre there any transactions having merchants among destination accounts\
          other than the PAYMENT type? {}'.format(\
         (df.loc[df.nameDest.str.contains('M')].type != 'PAYMENT').any()))
```

```
Are there merchants among any originator accounts? False

Are there any transactio                                          on accounts other than the PAYMENT 
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

Conclusion: Among the account labels nameOrig and nameDest, for all transactions, the merchant prefix of 'M' occurs in an unexpected way.

Are there account labels common to fraudulent TRANSFERs and CASH_OUTs? From the data description, the modus operandi for committing fraud involves first making a TRANSFER to a (fraudulent) account which in turn conducts a CASH_OUT. CASH_OUT involves transacting with a merchant who pays out cash. Thus, within this two-step process, the fraudulent account would be both, the destination in a TRANSFER and the originator in a CASH_OUT. However, the data shows below that there are no such common accounts among fraudulent transactions. Thus, the data is not imprinted with the expected modus-operandi.

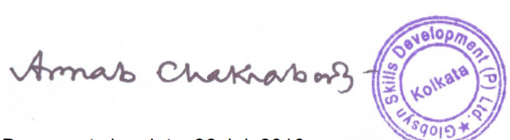
In [17]:
```
print('\nWithin fraudulent transactions, are there destinations for TRANSFERS \
that are also originators for CASH_OUTs? {}'.format(\
(dfFraudTransfer.nameDest.isin(dfFraudCashout.nameOrig)).any())) # False
dfNotFraud = df.loc[df.isFraud == 0]
```

```
Within fraudulent transactions, are there destinations for TRANSFERS that are also originators
```

Could destination accounts for fraudulent TRANSFERs originate CASHOUTs that are not detected and are labeled as genuine? It turns out there are 3 such accounts.

In [18]:
```
print('\nFraudulent TRANSFERs whose destination accounts are originators of \
genuine CASH_OUTs: \n\n{}'.format(dfFraudTransfer.loc[dfFraudTransfer.nameDest.\
isin(dfNotFraud.loc[dfNotFraud.type == 'CASH_OUT'].nameOrig.drop_duplicates())]))
```

```
Fraudulent TRANSFERs whose destination accounts are originators of genuine CASH_OUTs:

         step      type      amount     nameOrig  oldbalanceOrg  \
1030443    65  TRANSFER  1282971.57  C1175896731     1282971.57
6039814   486  TRANSFER   214793.32  C2140495649      214793.32
6362556   738  TRANSFER   814689.88  C2029041842      814689.88

         newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest  isFraud  \
1030443             0.0  C1714931087             0.0             0.0        1
6039814             0.0   C423543548             0.0             0.0        1
6362556             0.0  C1023330867             0.0             0.0        1

         isFlaggedFraud
1030443               0
6039814               0
6362556               0
```



Document sign date :22 Jul, 2018

However, 2 out of 3 of these accounts first make a genuine CASH_OUT and only later (as evidenced by the time step) receive a fraudulent TRANSFER. Thus, fraudulent transactions are not indicated by the nameOrig and nameDest features.

```
In [19]: print('\nFraudulent TRANSFER to C423543548 occured at step = 486 whereas \
         genuine CASH_OUT from this account occured earlier at step = {}'.format(\
         dfNotFraud.loc[(dfNotFraud.type == 'CASH_OUT') & (dfNotFraud.nameOrig == \
                                 'C423543548')].step.values))
```

```
Fraudulent TRANSFER to C423543548 occured at step = 486 whereas genuine CASH_OUT from this acc
```

Conclusion: Noting from section 2.3 above that the nameOrig and nameDest features neither encode merchant accounts in the expected way, below, we drop these features from the data since they are meaningless.

2 DATA CLEANING

From the exploratory data analysis (EDA), we know that fraud only occurs in

```
In [20]: df = df.loc[(df['type'].isin(['CASH_OUT', 'TRANSFER'])),:]
```

```
In [21]: df.drop(df.columns[[0,3,6,10]], axis=1, inplace=True)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:3697: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
  errors=errors)
```

```
In [22]: df.type=pd.factorize(df.type)[0]
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:4405: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
  self[name] = value
```

```
In [23]: df.head()
```

```
Out[23]:     type     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
         2      0     181.00          181.0             0.0             0.0
         3      1     181.00          181.0             0.0         21182.0
         15     1  229133.94        15325.0             0.0          5083.0
         19     0  215310.30          705.0             0.0         22425.0
         24     0  311685.89        10835.0             0.0          6267.0

             newbalanceDest  isFraud
         2             0.00        1
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

3	0.00	1
15	51513.44	0
19	0.00	0
24	2719172.89	0

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

NAIVE BAYES CLASSIFIER

1 NAIVE BAYES CLASSIFIER

In [42]:
```
import numpy as np
import  pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,precision_score,precision_recall_curve,au
from sklearn.model_selection import cross_val_score, train_test_split
```

In [13]:
```
# loading dataset
try:
    FraudTransactions=pd.read_csv('C:/Users\Raktim\Desktop\Python\PS_20174392719_1491
except:
    print('Database not able to load')
df=FraudTransactions
```

In [15]:
```
df = df.loc[(df['type'].isin(['CASH_OUT', 'TRANSFER'])),:] #selecting rows with type
df.drop(df.columns[[0,3,6,10]], axis=1, inplace=True) #droupping columns
df.type=pd.factorize(df.type)[0] #factorizing the type column
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:3697: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  errors=errors)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:4405: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self[name] = value
```

In [16]:
```
df.head()
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd Kolkata

Document sign date :22 Jul, 2018

Out[16]:
```
    type     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
2      0     181.00          181.0             0.0             0.0
3      1     181.00          181.0             0.0         21182.0
15     1  229133.94        15325.0             0.0          5083.0
19     0  215310.30          705.0             0.0         22425.0
24     0  311685.89        10835.0             0.0          6267.0

    newbalanceDest  isFraud
2             0.00        1
3             0.00        1
15        51513.44        0
19            0.00        0
24      2719172.89        0
```

In [30]:
```
y=df.isFraud
df_train,df_test,y_train,y_test=train_test_split(df.drop(['isFraud'],axis=1), y, test_
```

In [31]:
```
gnb = GaussianNB()
gnb.fit(df_train, y_train)
# making predictions on the testing set
y_pred = gnb.predict(df_test)
```

In [32]:
```
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[817014  11628]
 [  1485    996]]
```

In [33]:
```
print(classification_report(y_test,y_pred))
```

```
             precision    recall  f1-score   support

          0       1.00      0.99      0.99    828642
          1       0.08      0.40      0.13      2481

avg / total       1.00      0.98      0.99    831123
```

Arnab Chakraborty

Document sign date :22 Jul, 2018

Globsyn Skills Development (P) Ltd. Kolkata

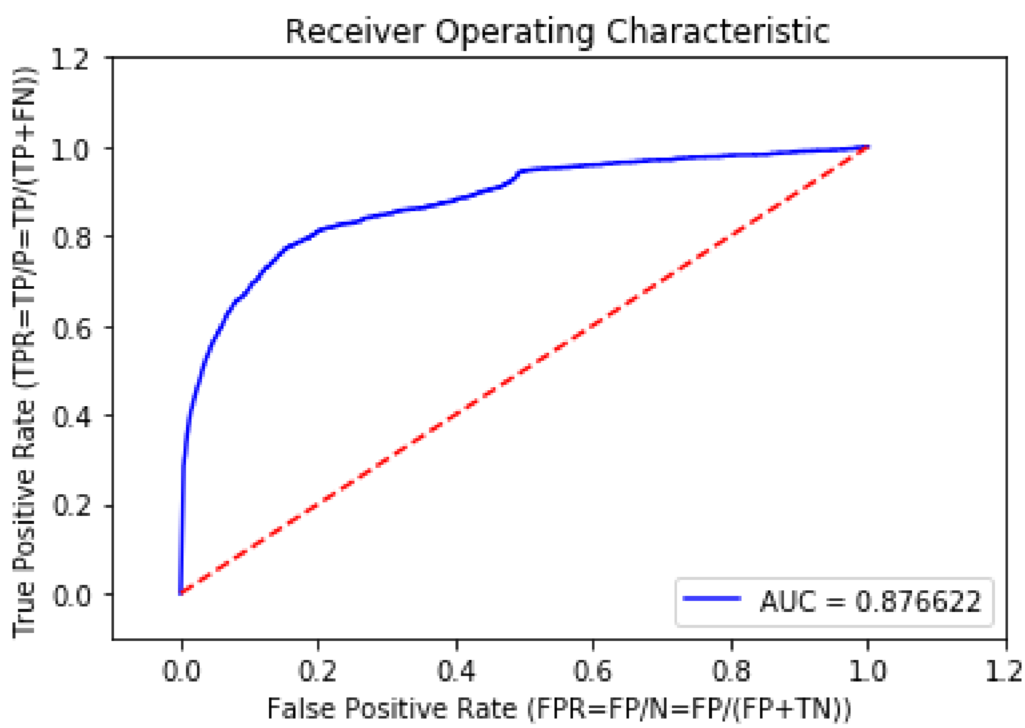
In [34]:
```
y_score = gnb.predict_proba(df_test)[:,1]
y_score
```

Out[34]:
```
array([2.46094364e-06, 1.26754865e-06, 1.69335171e-04, ...,
       1.00092893e-05, 1.94214334e-06, 2.43151286e-06])
```

In [35]:
```
false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_score)
```

In [36]:
```
roc_auc = auc(false_positive_rate, true_positive_rate)
```

In [37]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate (TPR=TP/P=TP/(TP+FN))')
plt.xlabel('False Positive Rate (FPR=FP/N=FP/(FP+TN))')
plt.show()
```



In [38]:
```
scores = cross_val_score(gnb,df, df.isFraud, cv=5)
print("Cross-validation scores: {}".format(scores))
```

```
Cross-validation scores: [0.97923055 0.98992748 0.98985529 0.96365874 0.98846378]
```

In [39]:
```
print("Average cross-validation score: {}".format(scores.mean()))
```

```
Average cross-validation score: 0.9822271695857202
```

Arnab Chakraborty

Skills Development (P) Ltd. Kolkata Globsyn

Document sign date :22 Jul, 2018

DECISION TREE CLASSIFIER

1 DECISION TREE

In [2]:
```
# implementing decision tree
# read data from dataset and import modules
import pandas as pd
binary = pd.read_csv('C:/Users\Raktim\Desktop\Python\PS_20174392719_1491204439457_log.
# print a few rows
binary.head()
```

Out[2]:
```
   step      type    amount     nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1   PAYMENT   9839.64  C1231006815       170136.0       160296.36
1     1   PAYMENT   1864.28  C1666544295        21249.0        19384.72
2     1  TRANSFER    181.00  C1305486145          181.0            0.00
3     1  CASH_OUT    181.00   C840083671          181.0            0.00
4     1   PAYMENT  11668.14  C2048537720        41554.0        29885.86

      nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155             0.0             0.0        0               0
1  M2044282225             0.0             0.0        0               0
2   C553264065             0.0             0.0        1               0
3    C38997010         21182.0             0.0        1               0
4  M1230701703             0.0             0.0        0               0
```

In [3]:
```
# drop a column
binary.drop("isFlaggedFraud",axis=1,inplace=True)
binary.drop("step",axis=1,inplace=True)
binary.drop("nameOrig",axis=1,inplace=True)
binary.drop("nameDest",axis=1,inplace=True)
# view few rows
binary.head()
```

Out[3]:
```
       type    amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
0   PAYMENT   9839.64       170136.0       160296.36             0.0
1   PAYMENT   1864.28        21249.0        19384.72             0.0
2  TRANSFER    181.00          181.0            0.00             0.0
3  CASH_OUT    181.00          181.0            0.00         21182.0
4   PAYMENT  116                                  36             0.0
```

```
   newbalanceDest  isFraud
0             0.0        0
1             0.0        0
2             0.0        1
3             0.0        1
4             0.0        0
```

In [4]:
```python
x=binary[(binary['type']=="TRANSFER") | (binary['type']=="CASH_OUT")]
x.head()
```

Out[4]:
```
        type     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
2   TRANSFER     181.00          181.0             0.0             0.0
3   CASH_OUT     181.00          181.0             0.0         21182.0
15  CASH_OUT  229133.94        15325.0             0.0          5083.0
19  TRANSFER  215310.30          705.0             0.0         22425.0
24  TRANSFER  311685.89        10835.0             0.0          6267.0

    newbalanceDest  isFraud
2             0.00        1
3             0.00        1
15        51513.44        0
19            0.00        0
24      2719172.89        0
```

In [5]:
```python
y=x["isFraud"].values
y
```

Out[5]:
```
array([1, 1, 0, ..., 1, 1, 1], dtype=int64)
```

In [7]:
```python
x.loc[x.type == 'TRANSFER', 'type'] = 0
x.loc[x.type == 'CASH_OUT', 'type'] = 1
x.type = x.type.astype(int)
x_cv=x.isFraud
del x['isFraud']
x.head()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
  self.obj[item] = s
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:4405: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
  self[name] = value
```

Arnab Chakraborty

Document sign date :22 Jul, 2018

Out[7]:
```
    type     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
2      0     181.00          181.0             0.0             0.0
3      1     181.00          181.0             0.0         21182.0
15     1  229133.94        15325.0             0.0          5083.0
19     0  215310.30          705.0             0.0         22425.0
24     0  311685.89        10835.0             0.0          6267.0

    newbalanceDest
2             0.00
3             0.00
15        51513.44
19            0.00
24      2719172.89
```

In [8]:
```python
# test and train samples
# now, splitting given dataset in to train and test datasets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.3,
                                random_state=176)
# print few rows of Train datasets
```

In [9]:
```python
# now constructing decision trees
# for constructing decision tree we are using CART algorithm
# (gini criteria).
from sklearn.tree import DecisionTreeClassifier
dt_train_gini = DecisionTreeClassifier(criterion = "gini", \
            random_state=100,max_depth=5,min_samples_leaf=5)
# train the model
dt_train_gini.fit(X_train,y_train)
```

Out[9]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=100,
            splitter='best')
```

In [10]:
```python
# to see the decision tree plot we shal use graphviz
from sklearn import tree
with open ("decision_tree1.txt","w") as f:
    f = tree.export_graphviz(dt_train_gini,out_file=f)
```

In [11]:
```python
# copy and paste the output file content at
# http://www.webgraphviz.com/ to visualize the graph
```

In [12]:
```python
y_predict=dt_train_gini.predict(X_test)
y_predict
```

Out[12]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [13]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_predict))
```

```
[[828619     29]
 [   831   1644]]
```

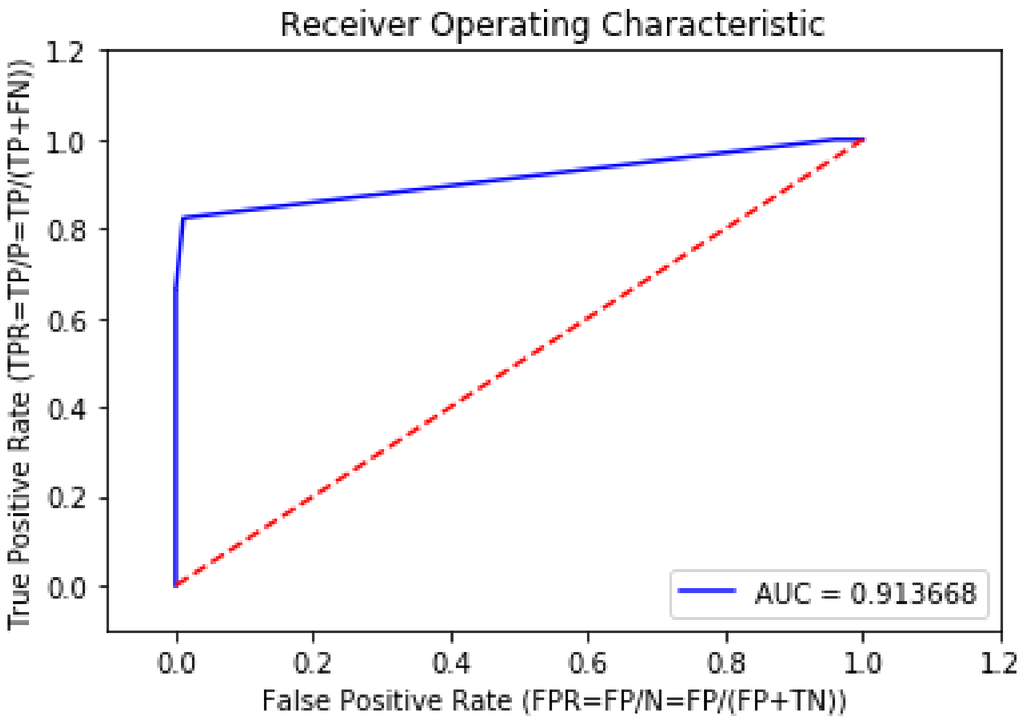
In [14]:
```
print(classification_report(y_test, y_predict))
```

```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00    828648
          1       0.98      0.66      0.79      2475

avg / total       1.00      1.00      1.00    831123
```

In [15]:
```
y_score=dt_train_gini.predict_proba(X_test)[:,1]
y_score
```

Out[15]:
```
array([0.00054897, 0.00054897, 0.00054897, ..., 0.00054897, 0.00054897,
       0.00054897])
```

In [19]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_score)
roc_auc = auc(false_positive_rate, true_positive_rate)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate (TPR=TP/P=TP/(TP+FN))')
plt.xlabel('False Positive Rate (FPR=FP/N=FP/(FP+TN))')
plt.show()
```




```
In [17]: from sklearn.model_selection import cross_val_score
         scores = cross_val_score(dt_train_gini,x,x_cv, cv=5)
         print("Cross-validation scores: {}".format(scores))
```

```
Cross-validation scores: [0.99873124 0.99892796 0.99902    0.99901458 0.99899112]
```

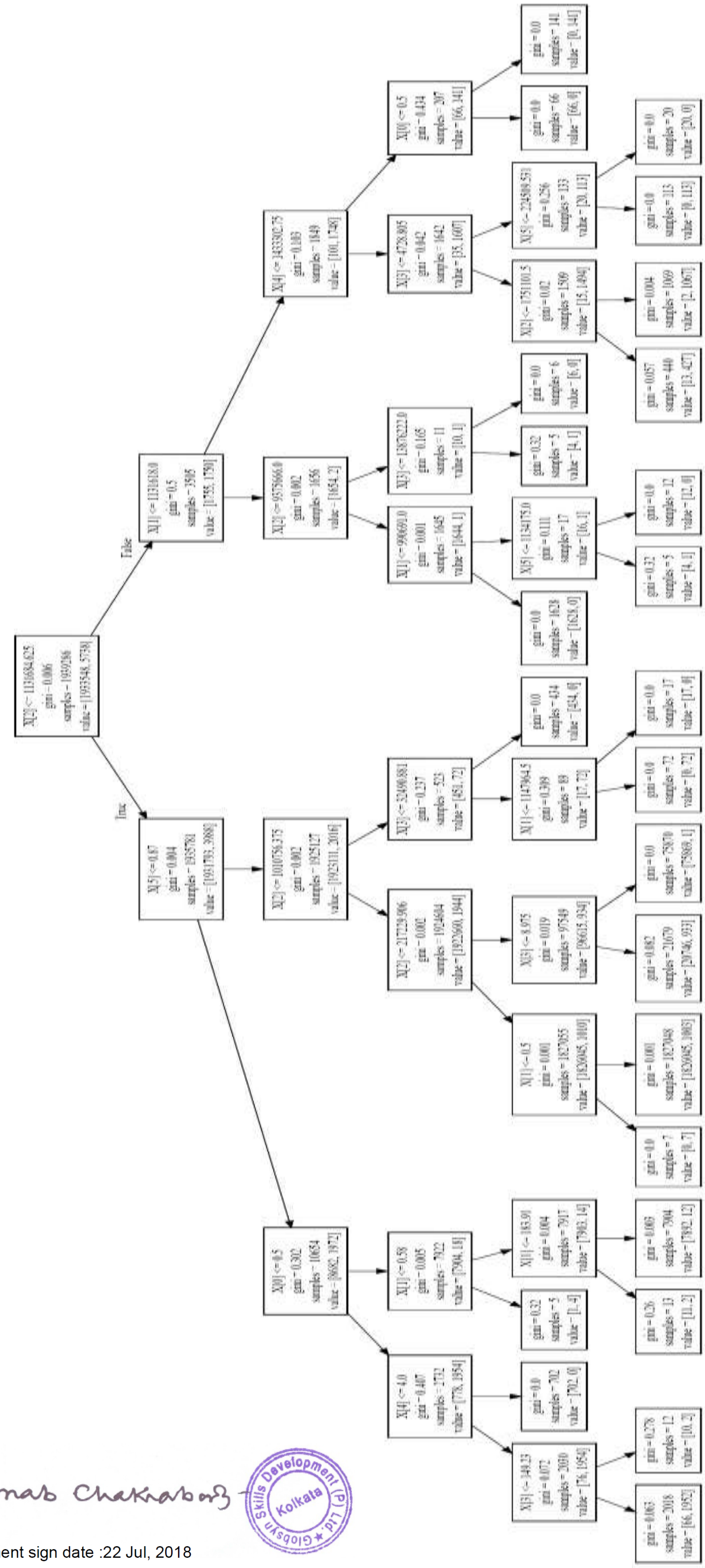
```
In [18]: print("Average cross-validation score: {}".format(scores.mean()))
```

```
Average cross-validation score: 0.9989369801850521
```

Arnab Chakraborty

Document sign date :22 Jul, 2018

Globsyn Skills Development (P) Ltd. Kolkata



Arnab Chakraborty

Globsyn Skills Development (P) Ltd. ★ Kolkata

Document sign date :22 Jul, 2018

K NEAREST NEIGHBOUR CLASSIFIER

1 K NEAREST NEIGHBOUR CLASSIFIER

In [3]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, auc
```

In [4]:
```python
data=pd.read_csv('C:/Users\Raktim\Desktop\Python\PS_20174392719_1491204439457_log.csv'
data.head()
```

Out[4]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72
2	1	TRANSFER	181.00	C1305486145	181.0	0.00
3	1	CASH_OUT	181.00	C840083671	181.0	0.00
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	M1979787155	0.0	0.0	0	0
1	M2044282225	0.0	0.0	0	0
2	C553264065	0.0	0.0	1	0
3	C38997010	21182.0	0.0	1	0
4	M1230701703	0.0	0.0	0	0

In [5]:
```python
a=data[(data['type']=="TRANSFER") | (data['type']=="CASH_OUT")]
a.head()
```

Out[5]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig
2	1	TRANSFER	181.00	C1305486145	181.0	0.0
3	1	CASH_OUT	181.00	C840083671	181.0	0.0
15	1	CASH_OUT	229133.94	C905080434	15325.0	0.0
19	1	TRANSFE[illegible]	[illegible]	[illegible]	705.0	0.0
24	1	TRANSFE[illegible]	[illegible]	[illegible]	[illegible]835.0	0.0

Arnab Chakraborty

Globsyn Skills Development (P) Ltd ★ Kolkata

Document sign date :22 Jul, 2018

```
       nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
2    C553264065             0.0            0.00        1               0
3     C38997010         21182.0            0.00        1               0
15   C476402209          5083.0        51513.44        0               0
19  C1100439041         22425.0            0.00        0               0
24   C932583850          6267.0      2719172.89        0               0
```

```
In [6]: y=a["isFraud"]
        y.head()
```

```
Out[6]: 2     1
        3     1
        15    0
        19    0
        24    0
        Name: isFraud, dtype: int64
```

```
In [7]: x=a.drop(["nameOrig","nameDest","isFraud","isFlaggedFraud"],axis=1)
        x.head()
```

```
Out[7]:     step      type     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
        2      1  TRANSFER     181.00          181.0             0.0             0.0
        3      1  CASH_OUT     181.00          181.0             0.0         21182.0
        15     1  CASH_OUT  229133.94        15325.0             0.0          5083.0
        19     1  TRANSFER  215310.30          705.0             0.0         22425.0
        24     1  TRANSFER  311685.89        10835.0             0.0          6267.0

            newbalanceDest
        2             0.00
        3             0.00
        15        51513.44
        19            0.00
        24      2719172.89
```

```
In [8]: scale=StandardScaler()
        x_to_scale=np.array(pd.DataFrame(x,columns=["amount","oldbalanceOrg","newbalanceOrig","
        X_scaled=scale.fit_transform(x_to_scale)
        X_scaled
```

```
Out[8]: array([[-0.35746665, -0.18884736, -0.10638868, -0.40315492, -0.43825939],
               [-0.35746665, -0.18884736, -0.10638868, -0.39814208, -0.43825939],
               [-0.09957563, -0.12859074, -0.10638868, -0.401952  , -0.42724516],
               ...,
               [ 6.75145759, 24.9229649 , -0.10638868, -0.38694665,  0.92584427],
               [ 0.59976648,  3.19251638, -0.10638868, -0.40315492, -0.43825939],
               [ 0.59976648,  3.19251638, -0.10638868,  1.13749648,  1.1354243 ]])
```

```
In [9]: x.loc[x.type == 'TRANSFER', 'type'] = 0
        x.loc[x.type == 'CASH_OUT', 'type'] = 1
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

```
x.type = x.type.astype(int)
x.head()
```

```
Out[9]:     step  type     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
        2      1     0     181.00          181.0             0.0             0.0
        3      1     1     181.00          181.0             0.0         21182.0
        15     1     1  229133.94        15325.0             0.0          5083.0
        19     1     0  215310.30          705.0             0.0         22425.0
        24     1     0  311685.89        10835.0             0.0          6267.0

            newbalanceDest
        2             0.00
        3             0.00
        15        51513.44
        19            0.00
        24      2719172.89
```

```
In [10]: x=x.reset_index(drop=True)
         x.head()
```

```
Out[10]:     step  type     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
         0      1     0     181.00          181.0             0.0             0.0
         1      1     1     181.00          181.0             0.0         21182.0
         2      1     1  229133.94        15325.0             0.0          5083.0
         3      1     0  215310.30          705.0             0.0         22425.0
         4      1     0  311685.89        10835.0             0.0          6267.0

             newbalanceDest
         0             0.00
         1             0.00
         2         51513.44
         3             0.00
         4       2719172.89
```

```
In [11]: x_type=x['type']
         x_type.head()
```

```
Out[11]: 0    0
         1    1
         2    1
         3    0
         4    0
         Name: type, dtype: int32
```

Arnab Chakraborty

Document sign date :22 Jul, 2018

Globsyn Skills Development (P) Ltd Kolkata

```
In [12]: #dummy=pd.DataFrame(pd.get_dummies(x["type"]))
         #dummy=dummy.set_index(np.arange(0,2770409))
         #dummy
```

```
In [13]: X_scaled_df=pd.DataFrame(X_scaled, columns=["amount","oldbalanceOrg","newbalanceOrig"
         #X_final=X_scaled_df.join(x_type, how='outer')
```

```
#X_final=X_final.values
#X_final
```

In [14]:
```
X_final_inner=X_scaled_df.join(x_type, how='inner')
#X_final=X_final.values
X_final_inner.head()
```

Out[14]:
```
     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  newbalanceDest  \
0 -0.357467      -0.188847       -0.106389       -0.403155       -0.438259
1 -0.357467      -0.188847       -0.106389       -0.398142       -0.438259
2 -0.099576      -0.128591       -0.106389       -0.401952       -0.427245
3 -0.115146      -0.186762       -0.106389       -0.397848       -0.438259
4 -0.006590      -0.146456       -0.106389       -0.401672        0.143134

   type
0     0
1     1
2     1
3     0
4     0
```

In [15]:
```
X_train,X_test,y_train,y_test=train_test_split(X_final_inner,y,test_size=0.50,random_s
```

In [16]:
```
knn=KNeighborsClassifier()
```

In [17]:
```
knn.fit(X_train,y_train)
```

Out[17]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

In [18]:
```
y_predict=knn.predict(X_test)
```

In [19]:
```
knn.score(X_test,y_test)
```

Out[19]:
```
0.9990694518139914
```

In [20]:
```
print(confusion_matrix(y_test,y_predict))
```

```
[[1380774     305]
 [    984    3142]]
```

In [21]:
```
print(classification_report(y_test, y_predict))
```

```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00   1381079
          1       0.91      0.76      0.83      4126

avg / total       1.00      1.00      1.00   1385205
```

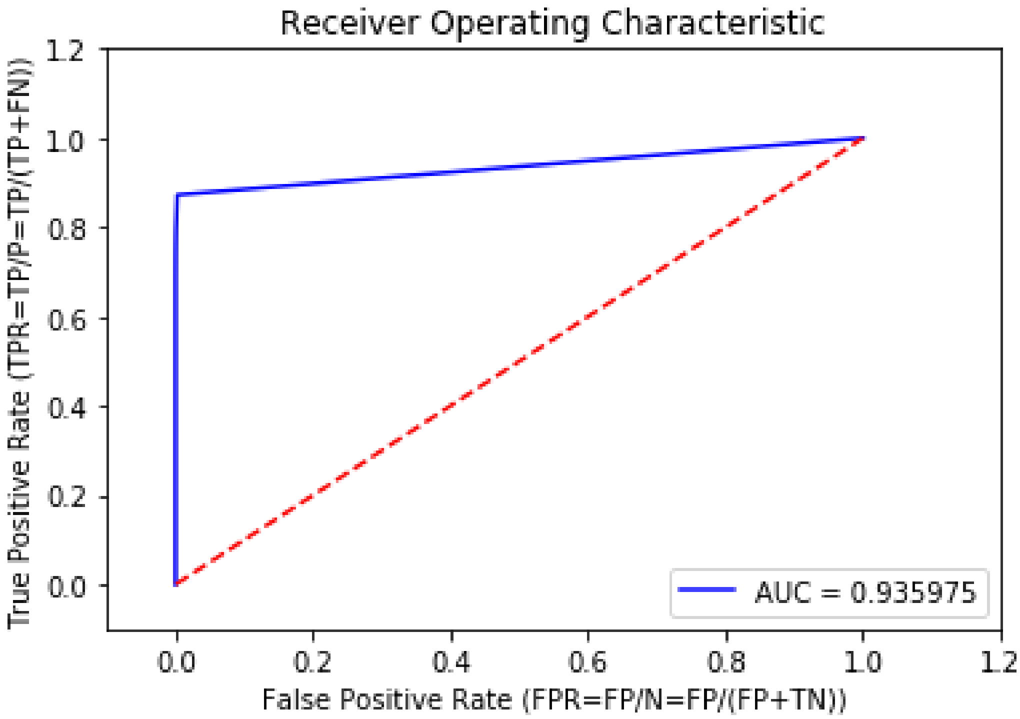
Arnab Chakraborty

Globsyn Skills Development (P) Ltd Kolkata

Document sign date :22 Jul, 2018

In [22]:
```
y_predict_proba=knn.predict_proba(X_test)[:,1]
```

In [23]:
```
false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_predict_proba
roc_auc = auc(false_positive_rate, true_positive_rate)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate (TPR=TP/P=TP/(TP+FN))')
plt.xlabel('False Positive Rate (FPR=FP/N=FP/(FP+TN))')
plt.show()
```



LOGISTIC REGRESSION CLASSIER

1 LOGISTIC REGRESSION

In [49]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, precision_score, precision_recall_curve
from sklearn.model_selection import cross_val_score,train_test_split
```

In [50]:
```
# loading dataset
try:
    FraudTransactions=pd.read_csv('C:/Users\Raktim\Desktop\Python\PS_20174392719_14912
except:
    print('Database not able to load')
df=FraudTransactions
```

In [51]:
```
df = df.loc[(df['type'].isin(['CASH_OUT', 'TRANSFER'])),:] #selecting rows with type
df.drop(df.columns[[0,3,6,10]], axis=1, inplace=True)      #droupping columns
df.type=pd.factorize(df.type)[0]                            #factorizing the type colu
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:3697: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  errors=errors)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:4405: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self[name] = value
```

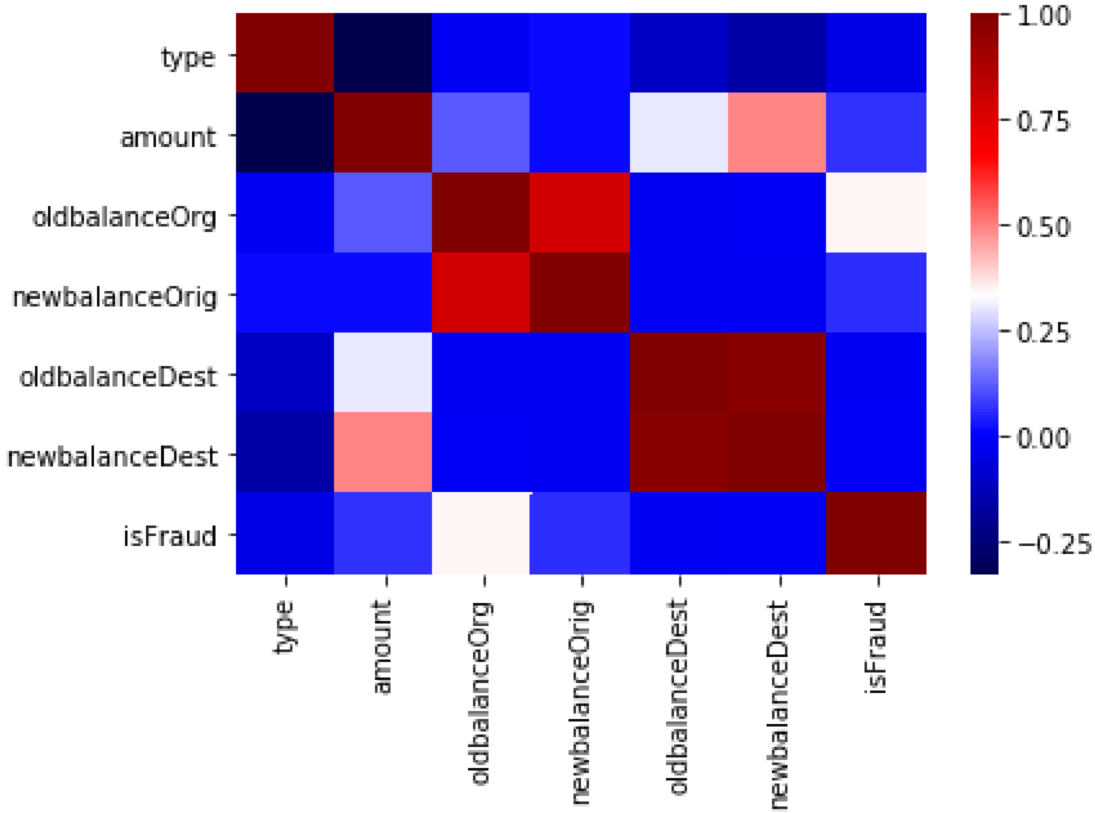
In [52]:
```
df.head()
```

Out[52]:

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	\
2	0	181.00	181.0	0.0	0.0	
3	1	181.00	181.0	0.0	21182.0	
15	1	229133.94	15325.0	0.0	5083.0	
19	0	215310.30	705.0	0.0	22425.0	
24	0	311685.89	10835.0	0.0	6267.0	

	newbalanceDest	isFraud
2	0.00	1
3	0.00	1
15	51513.44	0
19	0.00	0
24	2719172.89	0

In [53]:
```
sns.heatmap(df.corr(),cmap='seismic')
plt.show()
```



In [54]:
```
y=df.isFraud
df_train, df_test, y_train, y_test = train_test_split(df.drop(['isFraud'],axis=1), y,
```

In [55]:
```
classifier = Lo
classifier.fit(
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd ★ Kolkata

Document sign date :22 Jul, 2018

Out[55]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                    penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
                    verbose=0, warm_start=False)

In [ ]:

In [56]:
```
y_pred = classifier.predict(df_test)
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[551418   1005]
 [   465   1194]]
```

In [57]:
```
print(classification_report(y_test,y_pred))
```

```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00    552423
          1       0.54      0.72      0.62      1659

avg / total       1.00      1.00      1.00    554082
```

In [58]:
```
y_score = classifier.predict_proba(df_test)[:,1]
y_score
```

Out[58]:
```
array([1.45913954e-009, 2.45048190e-017, 1.02053611e-256, ...,
       2.32004552e-004, 1.16777770e-021, 1.19721054e-028])
```

In [59]:
```
false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_score)
```

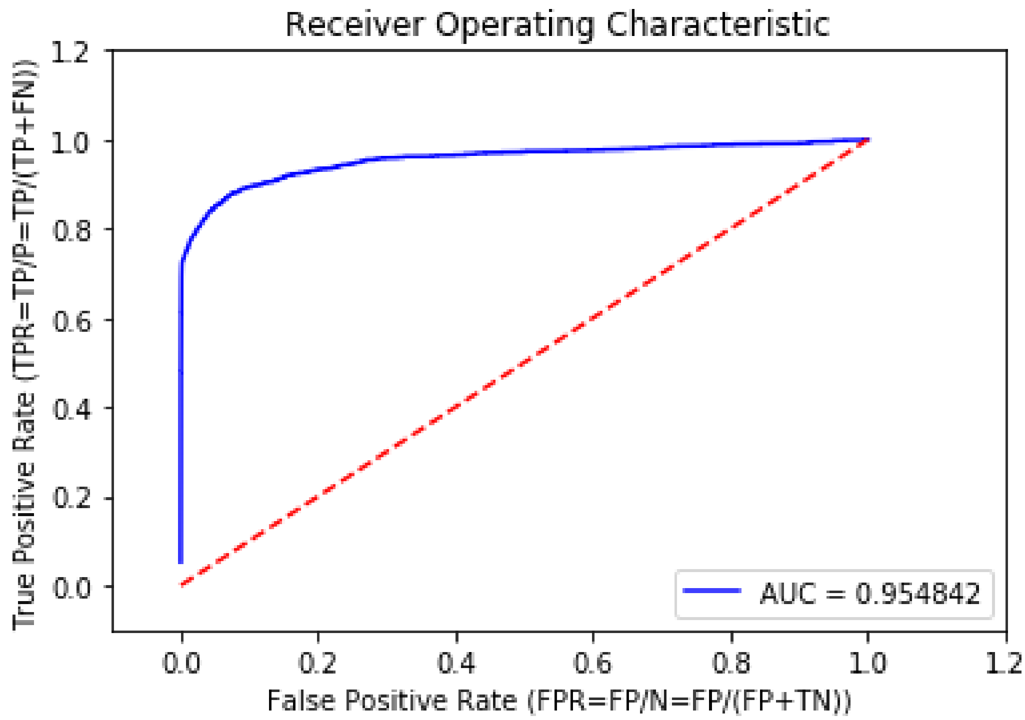
In [60]:
```
roc_auc = auc(false_positive_rate, true_positive_rate)
```

In [61]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',label='AUC = %f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate (TPR=TP/P=TP/(TP+FN))')
plt.xlabel('False Positive Rate (FPR=FP/N=FP/(FP+TN))')
plt.show()
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018




```
In [62]: scores = cross_val_score(classifier,df, df.isFraud, cv=5)
         print("Cross-validation scores: {}".format(scores))
```

```
Cross-validation scores: [0.99125402 0.9977368  0.99811219 0.99791547 0.99816814]
```

```
In [63]: print("Average cross-validation score: {}".format(scores.mean()))
```

```
Average cross-validation score: 0.9966373225730442
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

INFERENCE

USING RECALL VALUE

As such, specifically for this problem, we are interested in the recall score to capture the most fraudulent transactions. As we know, due to the imbalance of the data, many observations could be predicted as False Negatives, being, that we predict a normal transaction, but it is in fact a fraudulent one. Recall captures this.

Obviously, trying to increase recall, tends to come with a decrease of precision.

ALGORITHM	RECALL VALUE
NAÏVE BAYES	0.40
DECISION TREE	0.66
LOGISTIC REGRESSION	0.72
KNN	0.76

By looking at the Recall value we can say that KNN is the most suited for this dataset.

But at the same time, we must note that Recall value of Logistic Regression is 0.72 which is nearly as same as KNN. Thus, both the models can be equally suitable.

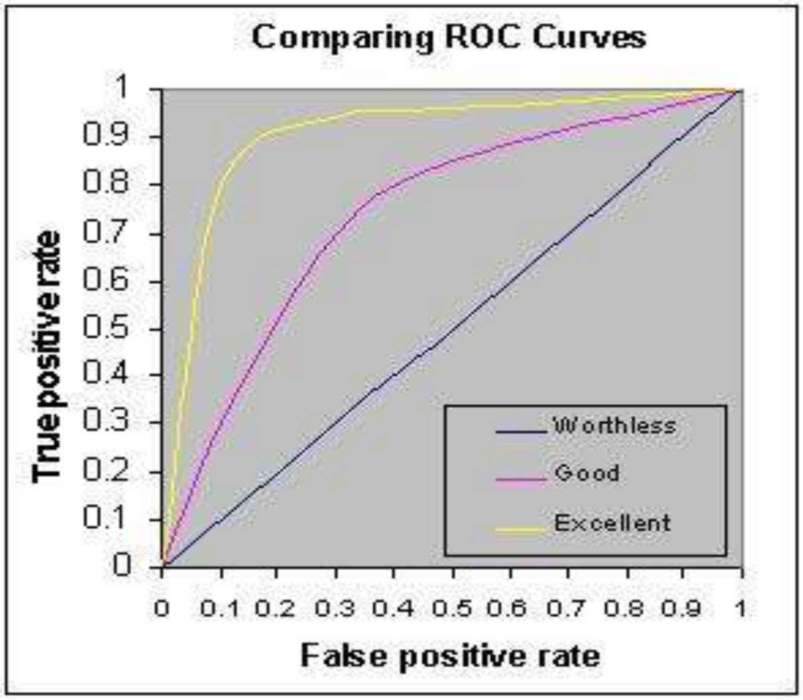
Arnab Chakraborty

Globsyn Skills Development (P) Ltd Kolkata

Document sign date :22 Jul, 2018

USING AREA UNDER RECEIVER OPERATING CHARACTERISTIC

The graph at right shows three ROC curves representing excellent, good, and worthless tests plotted on the same graph. The accuracy of the test depends on how well the test separates the group being tested into those with and without the disease in question. Accuracy is measured by the area under the ROC curve. An area of 1 represents a perfect test; an area of .5 represents a worthless test. A rough guide for classifying the accuracy of a diagnostic test is the traditional academic point system:



.90-1 = excellent (A)
.80-.90 = good (B)
.70-.80 = fair (C)
.60-.70 = poor (D)
.50-.60 = fail (F)

ALGORITHM	AU-ROC
NAÏVE BAYES	0.87
DECISION TREE	0.91
LOGISTIC REGRESSION	0.95
KNN	0.93

By looking at the AU-ROC values we find that Logistic Regression has a higher value than the others. But here also AU-ROC of both Logistic Regression and KNN are nearly same.Linear models are very fast to train, and also fast to predict. They scale to very large datasets and work well with sparse data. Another strength of linear models is that they make it relatively easy to understand how a prediction is made**. Therefore, we accept Logistic Regression as our best suited model**

Arnab Chakraborty

Globsyn Skills Development (P) Ltd Kolkata

Document sign date :22 Jul, 2018

FUTURE SCOPE OF IMPROVEMENTS

1. We can use **Artificial Neural Network** for getting better performance.
   - The Neural Networks are completely adaptive that is they are able to learn from patterns of legitimate behavior.
   - This makes the process of Neural Networks extremely fast and they can make decisions in real time.
   - It can be used in spam detection, image recognition, product recommendation, predictive analytics with proper changes in the model.
2. Use of classifiers such as **XGBoost**
   - Better performance on large training datasets. The regularization parameters really help create a sparse model compared to boosted regression and speed up computation.
3. Use of **Synthetic Minority Oversampling Technique (SMOTE)**
   - Synthetic Minority Oversampling Technique (SMOTE) is a very popular oversampling method that was proposed to improve random oversampling but its behavior on high-dimensional data has not been thoroughly investigated. Using SMOTE might lead to error in certain datasets depicting high False Positive Rate.
4. Use of Under Sampling
   - Under sample the dataset by creating a 50-50 ratio of randomly selecting 'x' amount of sample from majority class, with 'x' being the total number of records with the minority class
5. Use of StandardScaler
   - Standardize features by removing the mean and scaling to unit variance
   - Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using the transform method.

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

- Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual feature do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

6. Use of C parameter based Logistic Regression classifier
    - The parameter C is the the inverse of regularization strength in Logistic Regression and choose the best C value. This will help to increase the recall value.
7. Use of Feature Engineering to induce more derived featues and better understanding of the data.
8. Calculation of McFadden's R^2 since R^2 values for Logistic Regression are not credible.

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

BIBLIOGRAPHY

1. Data Science from Scratch: First Principles with Python
2. Hands-On Machine Learning with Scikit-Learn and Tensor Flow: Concepts, Tools, and Techniques to Build Intelligent Systems
3. Introduction to Machine Learning with Python: A Guide for Data Scientists
4. www.google.com
5. www.wikipedia.org
6. https://stackoverflow.com
7. https://www.slideshare.net/
8. Power Point Presentations from different sources

Arnab Chakraborty

Document sign date :22 Jul, 2018

APPENDIX

Decision_tree1.text

```
digraph Tree {
node [shape=box] ;
0 [label="X[2] <= 1131684.625\ngini = 0.006\nsamples =
1939286\nvalue = [1933548, 5738]"] ;
1 [label="X[5] <= 0.87\ngini = 0.004\nsamples = 1935781\nvalue
= [1931793, 3988]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[0] <= 0.5\ngini = 0.302\nsamples = 10654\nvalue =
[8682, 1972]"] ;
1 -> 2 ;
3 [label="X[4] <= 4.0\ngini = 0.407\nsamples = 2732\nvalue =
[778, 1954]"] ;
2 -> 3 ;
4 [label="X[3] <= 149.23\ngini = 0.072\nsamples = 2030\nvalue
= [76, 1954]"] ;
3 -> 4 ;
5 [label="gini = 0.063\nsamples = 2018\nvalue = [66, 1952]"] ;
4 -> 5 ;
6 [label="gini = 0.278\nsamples = 12\nvalue = [10, 2]"] ;
4 -> 6 ;
7 [label="gini = 0.0\nsamples = 702\nvalue = [702, 0]"] ;
3 -> 7 ;
8 [label="X[1] <= 0.58\ngini = 0.005\nsamples = 7922\nvalue =
[7904, 18]"] ;
2 -> 8 ;
9 [label="gini = 0.32\nsamples = 5\nvalue = [1, 4]"] ;
8 -> 9 ;
10 [label="X[1] <= 183.91\ngini = 0.004\nsamples = 7917\nvalue
= [7903, 14]"] ;
8 -> 10 ;
11 [label="gini = 0.26\nsamples = 13\nvalue = [11, 2]"] ;
10 -> 11 ;
12 [label="gini = 0.003\nsamples = 7904\nvalue = [7892, 12]"]
;
10 -> 12 ;
13 [label="X[2] <= 1010756.375\ngini = 0.002\nsamples =
1925127\nvalue = [1923111, 2016]"] ;
1 -> 13 ;
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd Kolkata

Document sign date :22 Jul, 2018

```
14 [label="X[2] <= 217229.906\ngini = 0.002\nsamples =
1924604\nvalue = [1922660, 1944]"] ;
13 -> 14 ;
15 [label="X[1] <= 0.5\ngini = 0.001\nsamples = 1827055\nvalue
= [1826045, 1010]"] ;
14 -> 15 ;
16 [label="gini = 0.0\nsamples = 7\nvalue = [0, 7]"] ;
15 -> 16 ;
17 [label="gini = 0.001\nsamples = 1827048\nvalue = [1826045,
1003]"] ;
15 -> 17 ;
18 [label="X[3] <= 8.975\ngini = 0.019\nsamples = 97549\nvalue
= [96615, 934]"] ;
14 -> 18 ;
19 [label="gini = 0.082\nsamples = 21679\nvalue = [20746,
933]"] ;
18 -> 19 ;
20 [label="gini = 0.0\nsamples = 75870\nvalue = [75869, 1]"] ;
18 -> 20 ;
21 [label="X[3] <= 32490.881\ngini = 0.237\nsamples =
523\nvalue = [451, 72]"] ;
13 -> 21 ;
22 [label="X[1] <= 1147964.5\ngini = 0.309\nsamples =
89\nvalue = [17, 72]"] ;
21 -> 22 ;
23 [label="gini = 0.0\nsamples = 72\nvalue = [0, 72]"] ;
22 -> 23 ;
24 [label="gini = 0.0\nsamples = 17\nvalue = [17, 0]"] ;
22 -> 24 ;
25 [label="gini = 0.0\nsamples = 434\nvalue = [434, 0]"] ;
21 -> 25 ;
26 [label="X[1] <= 1131618.0\ngini = 0.5\nsamples =
3505\nvalue = [1755, 1750]"] ;
0 -> 26 [labeldistance=2.5, labelangle=-45, headlabel="False"]
;
27 [label="X[2] <= 9375666.0\ngini = 0.002\nsamples =
1656\nvalue = [1654, 2]"] ;
26 -> 27 ;
28 [label="X[1] <= 990691.0\ngini = 0.001\nsamples =
1645\nvalue = [1644, 1]"] ;
27 -> 28 ;
29 [label="gini = 0.0\nsamples = 1628\nvalue = [1628, 0]"] ;
28 -> 29 ;
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

```
30 [label="X[5] <= 1134175.0\ngini = 0.111\nsamples =
17\nvalue = [16, 1]"] ;
28 -> 30 ;
31 [label="gini = 0.32\nsamples = 5\nvalue = [4, 1]"] ;
30 -> 31 ;
32 [label="gini = 0.0\nsamples = 12\nvalue = [12, 0]"] ;
30 -> 32 ;
33 [label="X[3] <= 13876222.0\ngini = 0.165\nsamples =
11\nvalue = [10, 1]"] ;
27 -> 33 ;
34 [label="gini = 0.32\nsamples = 5\nvalue = [4, 1]"] ;
33 -> 34 ;
35 [label="gini = 0.0\nsamples = 6\nvalue = [6, 0]"] ;
33 -> 35 ;
36 [label="X[4] <= 1433302.75\ngini = 0.103\nsamples =
1849\nvalue = [101, 1748]"] ;
26 -> 36 ;
37 [label="X[3] <= 4728.805\ngini = 0.042\nsamples =
1642\nvalue = [35, 1607]"] ;
36 -> 37 ;
38 [label="X[2] <= 1751101.5\ngini = 0.02\nsamples =
1509\nvalue = [15, 1494]"] ;
37 -> 38 ;
39 [label="gini = 0.057\nsamples = 440\nvalue = [13, 427]"] ;
38 -> 39 ;
40 [label="gini = 0.004\nsamples = 1069\nvalue = [2, 1067]"] ;
38 -> 40 ;
41 [label="X[5] <= 224509.531\ngini = 0.256\nsamples =
133\nvalue = [20, 113]"] ;
37 -> 41 ;
42 [label="gini = 0.0\nsamples = 113\nvalue = [0, 113]"] ;
41 -> 42 ;
43 [label="gini = 0.0\nsamples = 20\nvalue = [20, 0]"] ;
41 -> 43 ;
44 [label="X[0] <= 0.5\ngini = 0.434\nsamples = 207\nvalue =
[66, 141]"] ;
36 -> 44 ;
45 [label="gini = 0.0\nsamples = 66\nvalue = [66, 0]"] ;
44 -> 45 ;
46 [label="gini = 0.0\nsamples = 141\nvalue = [0, 141]"] ;
44 -> 46 ;
```

Arnab Chakraborty

Globsyn Skills Development (P) Ltd. Kolkata

Document sign date :22 Jul, 2018

globsyn finishing school

Skill India
कौशल भारत - कुशल भारत

Certificate Id: **WB/GFS/SS18/40044**

Certificate

This is to Certify that

Mr./Mrs./Ms. Raktim Mukhopadhyay

Of

Government College Of Engineering & Ceramic Technology

has successfully completed program in

Machine Learning with Python *(sub trade/s)*

Date: 16.08.2018

Authorised Signatory

THANK YOU