Fraud Detection in Mobile Money Transactions

The Problem

The aim of this project is to use computer simulation for fraud detection in mobile money financial transactions. Fraud prevention is becoming a critical driver for the financial services industry. In the recent years, due to an increase in use of new technologies such as cloud and mobile financial services, the fraud problem has been intensified. Therefore achieving an accurate and less intrusive fraud detection system is crucial and financial services institutions are increasingly investing in algorithms and data analytics technology to spot and combat fraud.

Clients

The primary client for this project would be banks and financial services that provide mobile money transactions. However, financial fraud is a problem which affects the finance industry, government, corporate sectors, and ordinary consumers, and therefore identifying and preventing fraud can be beneficial for many different clients.

Current Dataset

Currently, there is a lack of public research into the detection of fraud. One important reason is shortage of transaction data due to confidentiality issues. Due to this problem, a synthetic dataset generated using the simulator called PaySim is used in this project. PaySim uses aggregated data from the private dataset to generate a synthetic dataset that resembles the normal operation of transactions and injects malicious behaviour to later evaluate the performance of fraud detection methods. The data is described in details in the following PhD thesis: http://urn.kb.se/resolve?urn=urn:nbn:se:bth-12932

Description:

The Paysim synthetic dataset of mobile money transactions available on Kaggle is used for this project. The transaction data is presented in different steps, each step representing an hour of simulation. The raw data consists of 11 columns and about 6362620 rows. The description of each column is as follows:

step: Maps a unit of time in the real world. In this case, one step is one hour of time. Total steps are 744 (30 days simulation).
type: CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.
amount: Amount of the transaction in local currency.
nameOrig: Customer who started the transaction.
oldbalanceOrig: Initial balance before the transaction.
newbalanceOrig: New balance after the transaction.
nameDest: Customer who is the recipient of the transaction.
oldbalanceDest: Initial balance recipient before the transaction. Note that there is no information for customers that start with M (Merchants).
newbalanceDest: new balance recipient after the transaction. Note that there is no information for customers that start with M (Merchants).
isFraud: This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behaviour 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: 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.

Transaction types are defined based on the following reference: http://bth.diva-portal.org/smash/get/diva2:955852/FULLTEXT06.pdf
CASH-IN is the process of increasing the balance of account by paying in cash to a merchant.
CASH-OUT is the opposite process of CASH-IN, it means to withdraw cash from a merchant which decreases the balance of the account.
DEBIT is similar process than CASH-OUT and involves sending the money from the mobile money service to a bank account.

PAYMENT is the process of paying for goods or services to merchants which decreases the balance of the account and increases the balance of the receiver.
TRANSFER is the process of sending money to another user of the service through the mobile money platform.

Data Wrangling

By looking at the legitimate versus fraudulent transactions, it can be seen that the number of fraudulent transactions are much lower than the legitimate ones. Therefore the data is highly imbalanced.

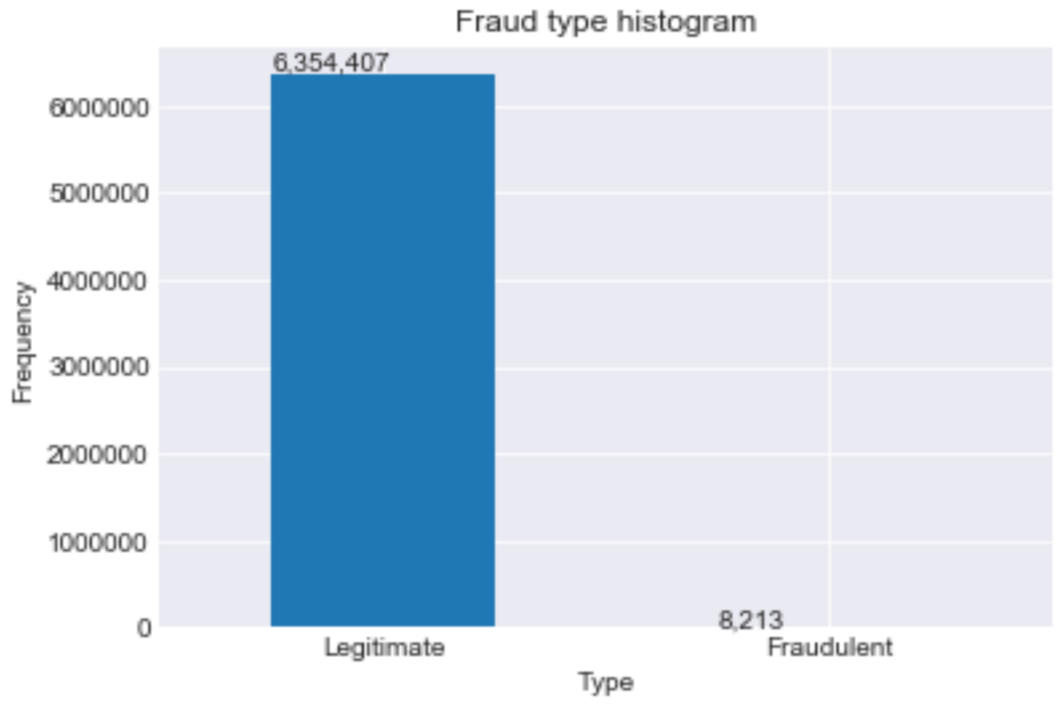
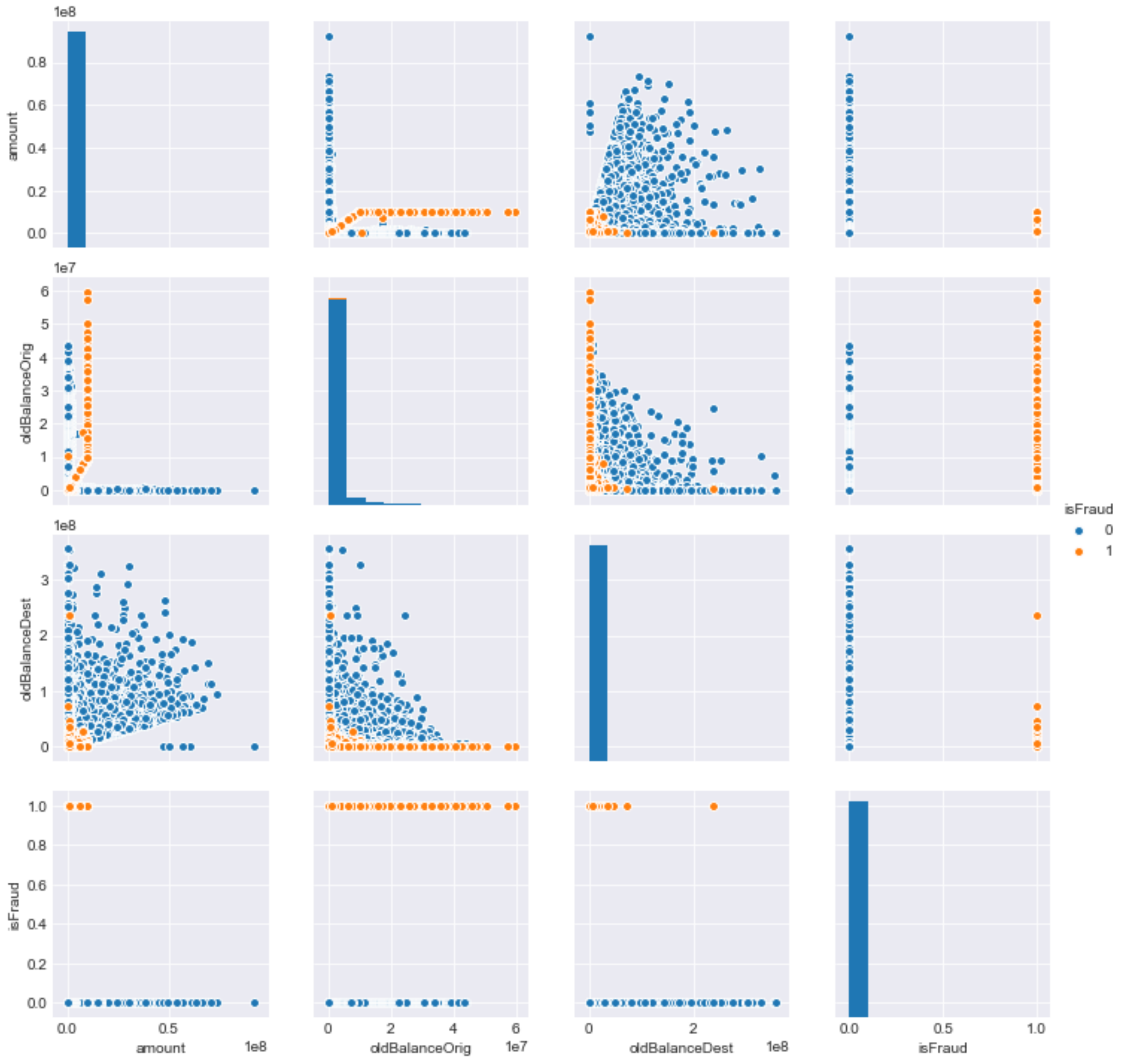
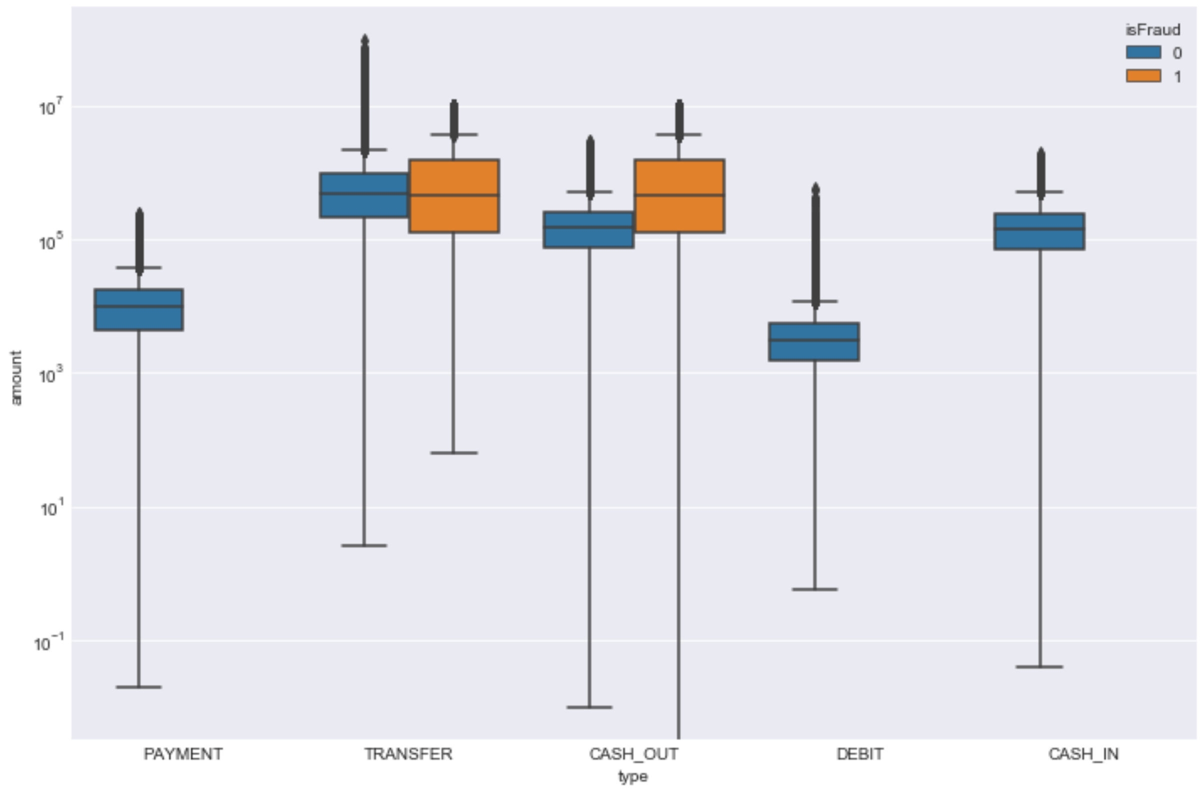


Figure below shows the pair plot for different features for both legitimate and fraudulent transactions.



Next we look at the amount variations for fraudulent and legitimate transactions. It can be seen that fraud only happens in TRANSFER and CASH-OUT transactions:

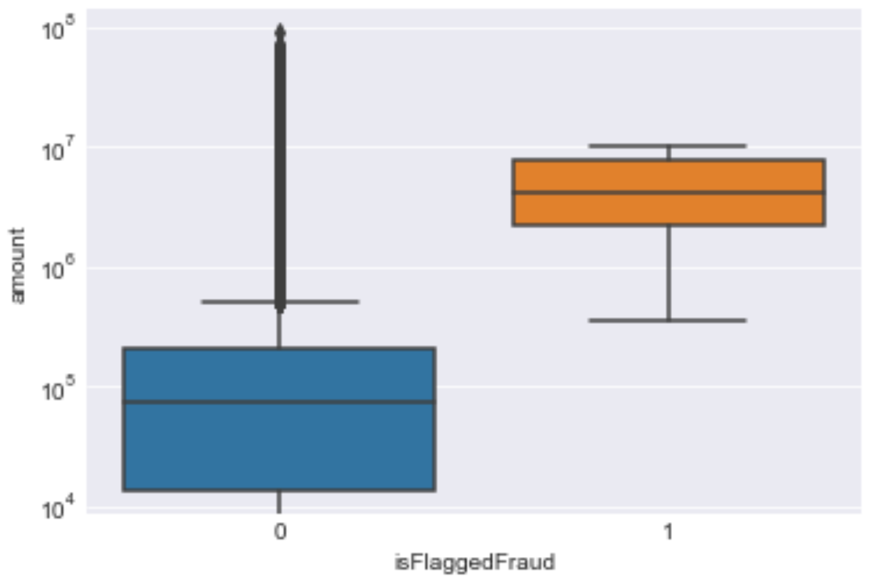


Number of TRANSFER fraudulent activities: 4097
Number of CASH-OUT fraudulent activities: 4116

It can also be seen that the amounts are higher for fraudulent transactions, in both TRANSFER and CASH-OUT.

Checking IsFlaggedFraud Feature

Next we look at IsFlaggedFraud feature. The amount for IsFlaggedFraud values of 0 and 1 is shown below. The amount for IsFlaggedFraud is higher than non Flagged amounts. Also the type of transactions in which isFlaggedFraud=1 is only TRANSFER.



As described in the features definitions, all the transactions which are "isFlaggedFraud" are the ones with transaction amounts higher than 200,000. We check to see if this holds for our data:

Number of isFlaggedFraud=1 cases where the amount if lower or equal to 200,000: 0
Number of TRANSFERs over 200,000 which are not flagged as isFlaggedFraud: 409094

So it seems that any TRANSFER over 200,000 is not necessarily flagged as isFlaggedFraud=1, which is in contrast with the definition.

Now we look at the correlation between isFraud and isFlaggedFraud:
Number of cases where isFlaggedFraud=1 but isFraud=0: 0
Number of cases where isFraud=1 but isFlaggedFraud=0: 8197

While whenever isFlaggedFraud=1, there is actually a fraud case, there are 8197 cases which are Fraud cases but isFlaggedFraud has failed to flag them as fraudulent. There are also only 16 TRANSFERs which are flagged as fraud.

Next we check the values and differences of oldBalanceDest and newBalanceDest for isFlaggedFraud and TRANSFER:

Sum of difference between new and old balance of destinations for flagged transfers: 0
Number of non-zero old balance of destinations for flagged transfers: 0

So any TRANSFER that has been flagged fraudulent, does not have any old or new destination balances. This can be due to suspension of the transfers because of the fraud flag. So destination balances do not give us any condition for flagging a transaction fraud.

By checking the number of times a customer with isFlaggedFraud TRANSFER had more than one transaction, it is observed that no customer with FlaggedFraud TRANSFER had more than one transaction. Also there is no case in which the origin and destination of a FlaggedFraud TRANSFER is the same person.

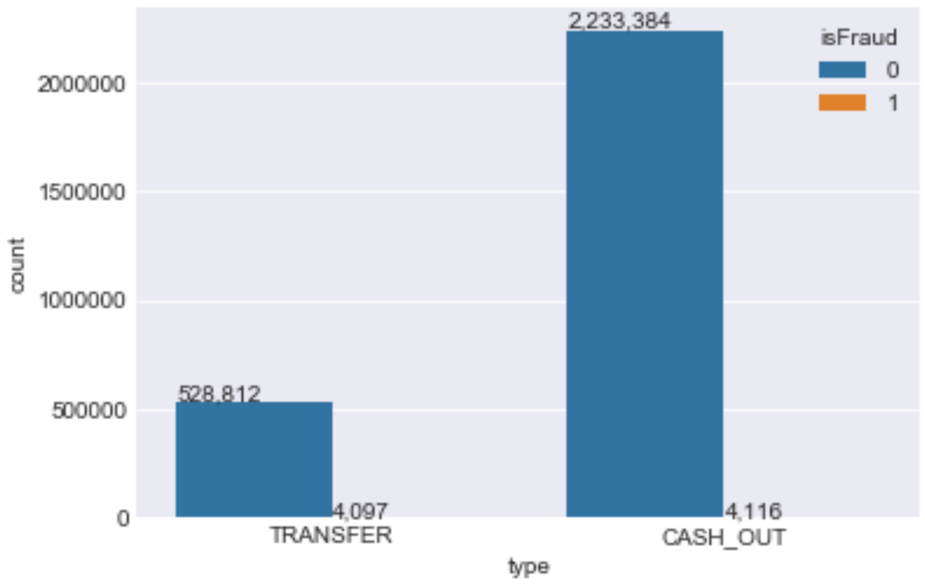
Looking at all the analysis above, it seems that isFlaggedFraud does not play any significant meaningful role in setting Fraud cases. Even though isFraud is set whenever isFlaggedFraud is set to true, isFlaggedFraud is on only for 16 TRANSFERs. Therefore, isFlaggedFraud column will be removed in this dataset.

Looking at NameDest feature

According to the definition of features mentioned above, a merchant should be involved in both CASH-IN and CASH-OUT transactions. However, by looking at the number of fraudulent cases involving merchants, one can realize that there is no record of balance from clients that start with M (Merchants). Since there are many CASH-OUT cases in fraudulent transactions, and we don't see any merchant involved in them, it can be concluded that the name columns are not defined properly and can be dropped.

Dropping other transaction types

Since there are only two types of transactions involved in a fraud (TRANSFER and CASH-OUT), we will delete the other rows.



Treating zero balances

There are lots of zero values in new and old destination balances even though the amount is not zero. These values may need to be replaced.

First we will take a look at the percentage of above condition for both fraudulent and legitimate transactions:
Percentage of fraudulent transactions where new and old destination balances are zero, even though the amount is not zero: 49.55 %
Percentage of legitimate transactions where new and old destination balances are zero, even though the amount is not zero: 0.06 %

Percentage of fraudulent transactions where new and old original balances are zero, even though the amount is not zero: 0.30 %
Percentage of legitimate transactions where new and old original balances are zero, even though the amount is not zero: 47.37 %

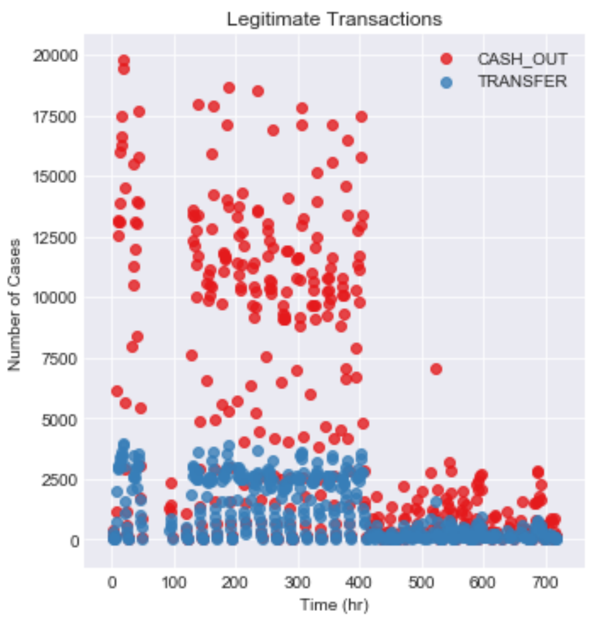
It is clear that the old and new destination balances being zero when amount is not zero happens much more often for fraudulent transactions. So it can be an indicator of the fraud. Therefore these values should not be replaced.

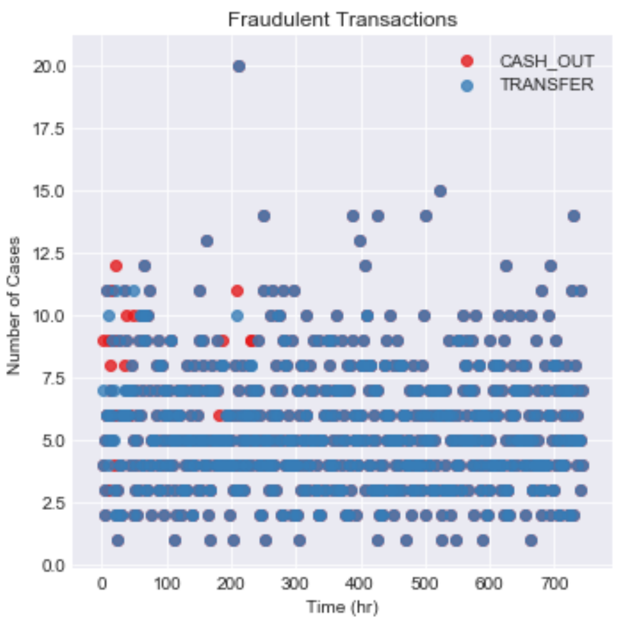
Also for original accounts, fraudulent transactions have much less percentage of new and old original balances being zero compared to legitimate transactions. For the same reason as above, we will not replace these numbers. Instead, we will create two new features calculating the error for both original and destination balances as follows:

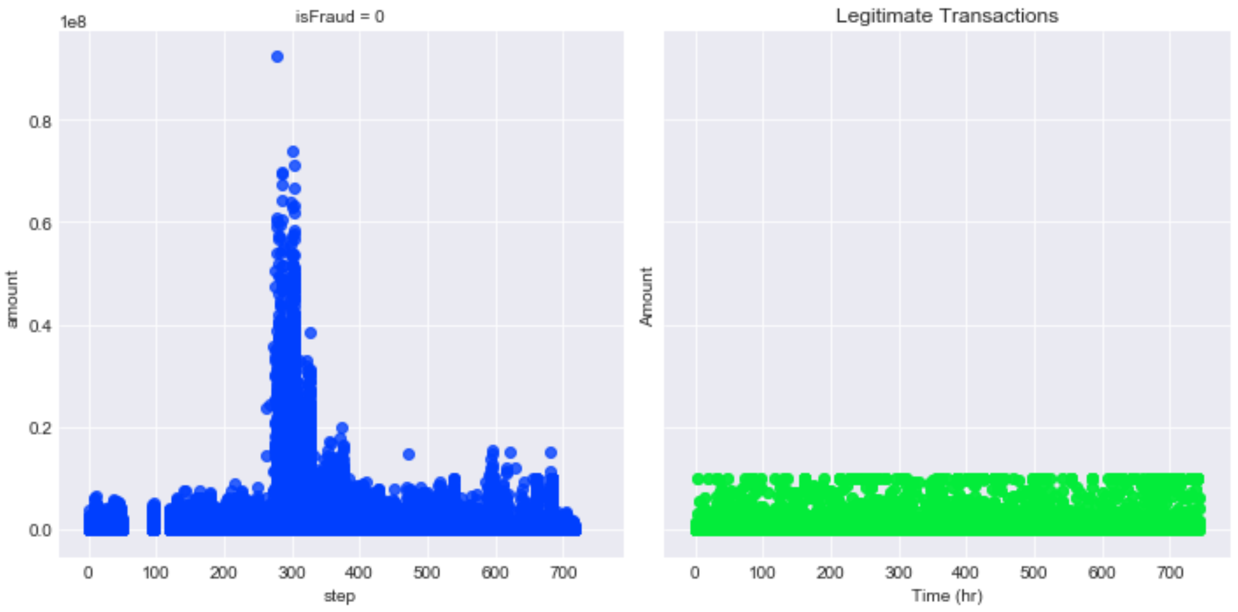
```
X_Fraud['errorBalanceOrig']=X_Fraud['newBalanceOrig']+X_Fraud['amount']-X_Fraud['oldBalanceOrig']
X_Leg['errorBalanceOrig']=X_Leg['newBalanceOrig']+X_Leg['amount']-X_Leg['oldBalanceOrig']
X_Fraud['errorBalanceDest']=X_Fraud['newBalanceDest']+X_Fraud['amount']-X_Fraud['oldBalanceDest']
X_Leg['errorBalanceDest']=X_Leg['newBalanceDest']+X_Leg['amount']-X_Leg['oldBalanceDest']
X['errorBalanceOrig']=X['newBalanceOrig']+X['amount']-X['oldBalanceOrig']
X['errorBalanceDest']=X['newBalanceDest']+X['amount']-X['oldBalanceDest']
```

Time Variations

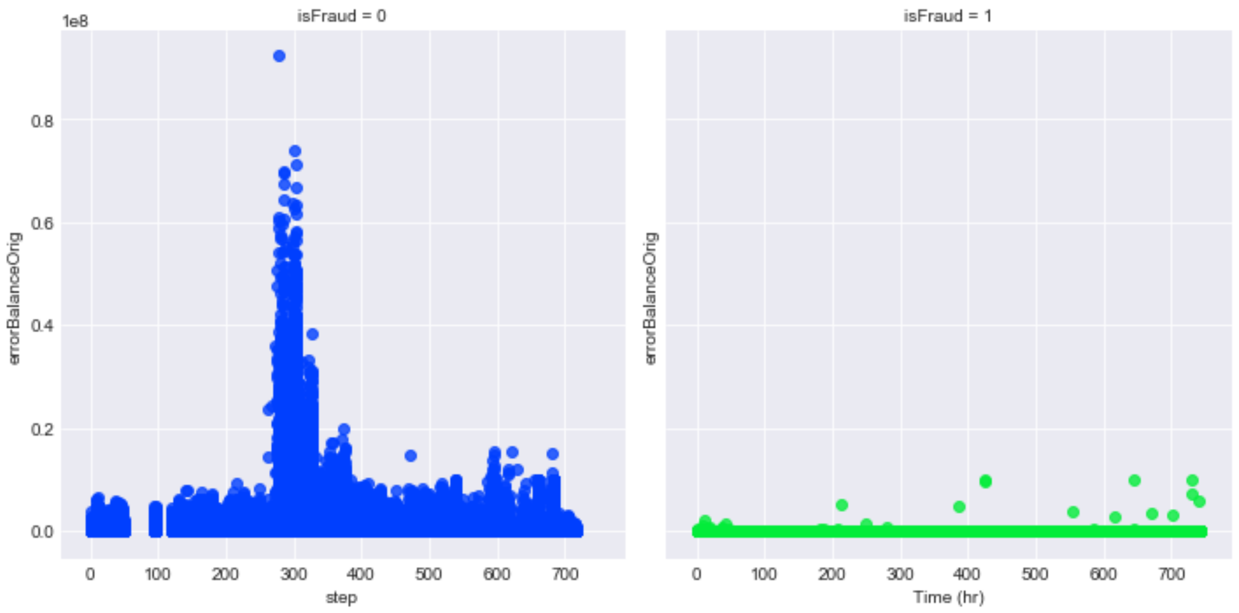
The changes in number of fraudulent and legitimate transactions over time are plotted next:



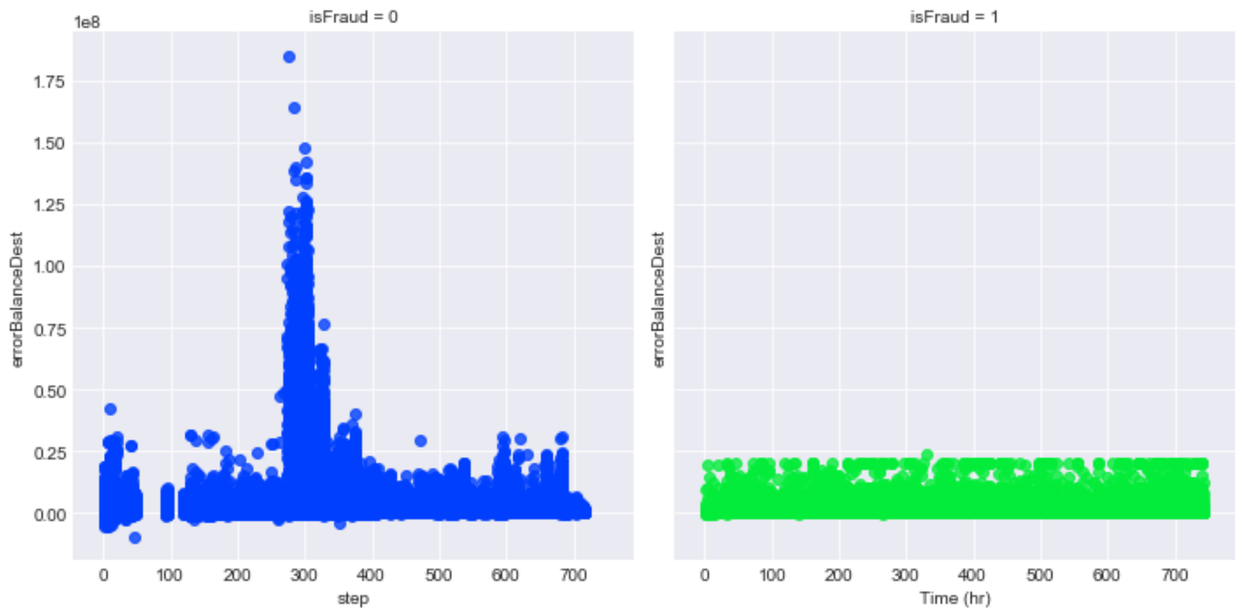




From the above figures, it is clear that for legitimate transactions, number of CASH-OUT is higher than TRANSFERs per hour, while for fraudulent transactions, CASH-OUT and TRANSFER are equal on many hours. Also, fraudulent cases are more distributed over time than legitimate transactions. For legitimate cases, both transactions drop after hour 400 (17 days).

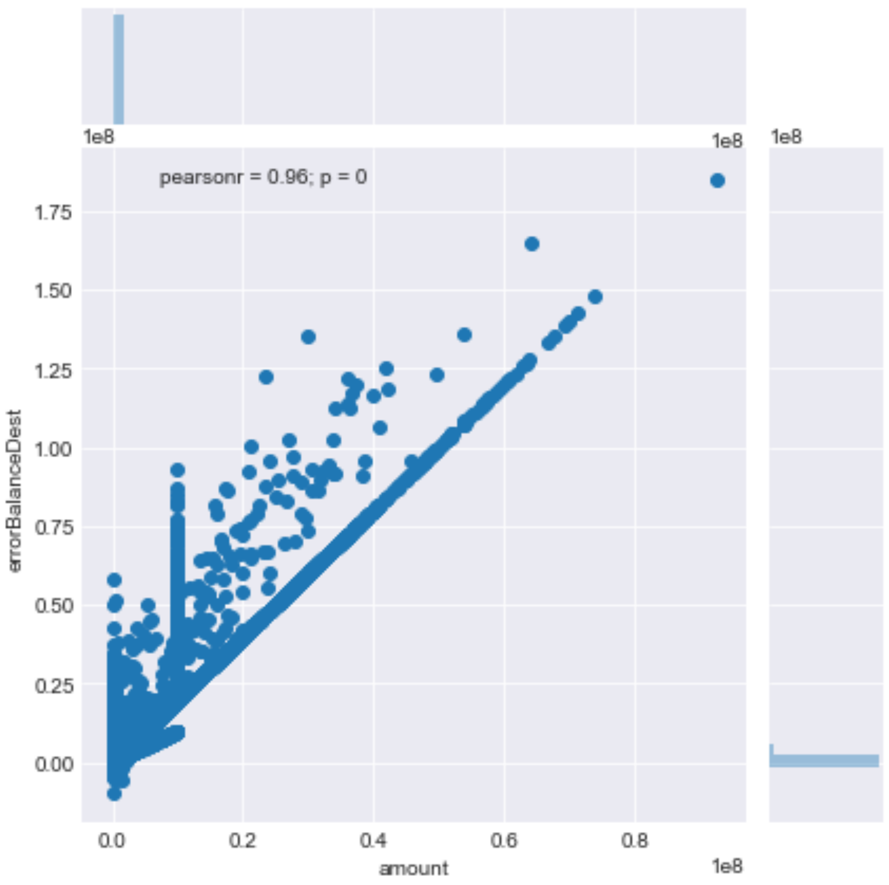


Amount variations over time is plotted below:



For legitimate transactions, there is a peak in amount at around 300hr, while for fraudulent cases, the amount is more evenly distributed.

The time variations of errorBalances are shown below. While the errorBalanceDest plots are very similar to amount, there is more clear distinction between fraudulent and legitimate cases for errorBalanceOrig values over time. This can also be seen on the correlation plot below.

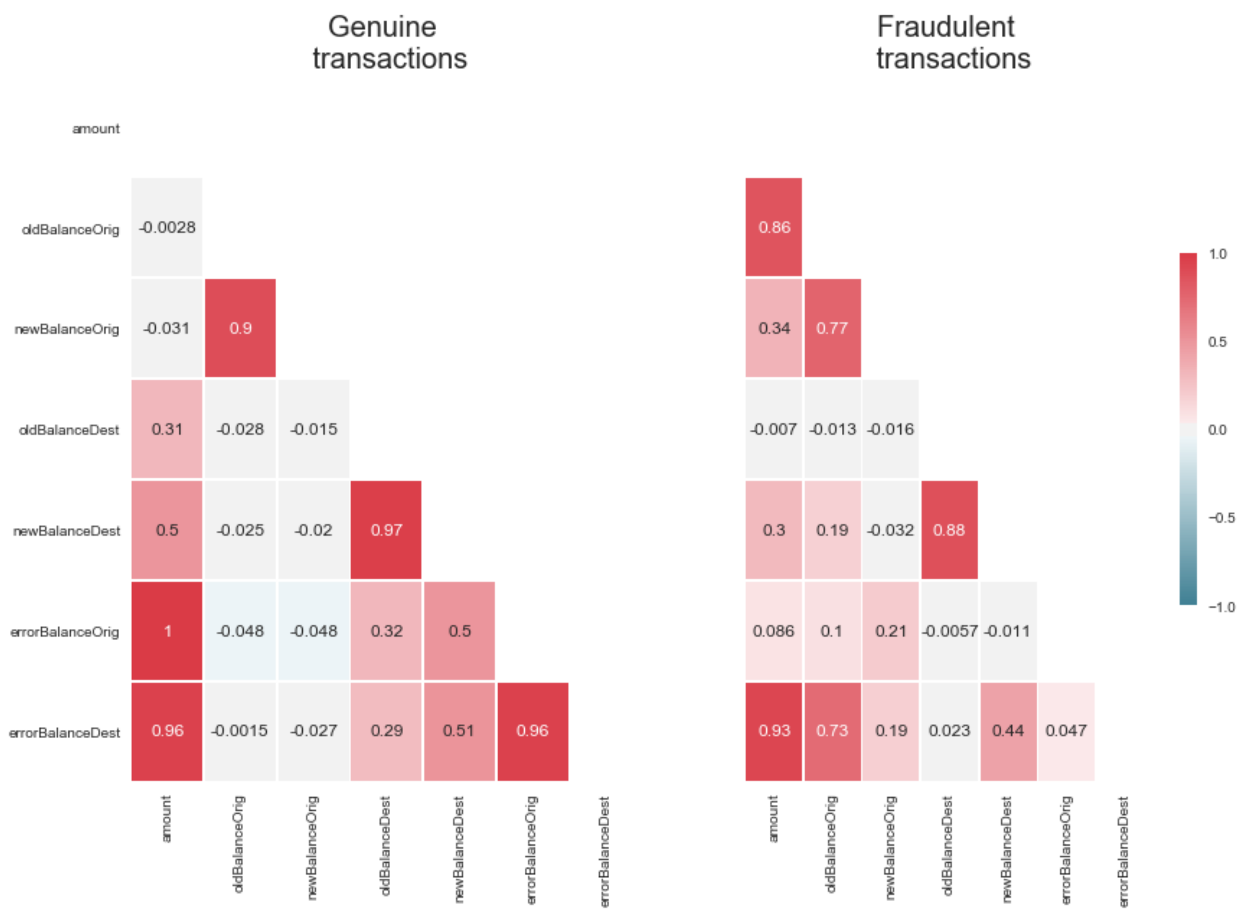


The respective correlation of different features is shown on the heat map below. There is a high correlation between oldBalanceOrig and amount in the fraudulent transactions, which cannot be observed in the legitimate cases.

Inferential Statistics

In this section, some inferential statistics techniques are applied for the fraud detection data, in order to check whether or not the fraudulent and legitimate transactions are statistically different in the amount of transactions. The Null hypothesis is that there is no difference in amount of transactions between fraudulent and legitimate cases. Alternative hypothesis is that this amount is statistically different in fraudulent and legitimate transactions.

In order to perform this statistical evaluation, the mean of amount for fraudulent and legitimate cases are calculated, which are 1467967.30 and 314115.49,



respectively. After calculating the standard deviation and size for each group, the standard error and z-score are obtained as follows:
standard error = 26534.76
z-score = 43.484

The p-value based on the above z-score is calculated as zero. Since the p-value is zero, we reject the null hypothesis. So there is a statistically significant difference between mean of amount for legitimate and fraudulent transactions.

Other Potential Datasets

Kaggle’s credit card fraud detection data:

The datasets contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, the original features and more background information about the data is not provided.

German credit fraud dataset:

This dataset classifies people described by a set of attributes as good or bad credit risks. It contains categorical and integer data with 20 features and 1000 instances.

Initial Findings

The main technical challenge in predicting fraud is the highly imbalanced distribution between legitimate and fraudulent classes in 6 million rows of data. Another deficiency of this data stems from the possible discrepancies in its description and some redundant column values. The goal of this project is to solve these issues by a detailed data exploration and wrangling followed by choosing a suitable machine-learning algorithm to deal with the skew. Supervised classification algorithms will be used to predict fraudulent transactions.

Machine Learning

The primary goal of the machine learning section is to identify the best algorithm to classify the fraudulent transactions. The following models are used for this study:

1. Random Forest Classifier

2. Support Vector Machine (SVM) classifier

3. Logistic Regression Classifier

Before starting fitting different models, some data preparation needs to be done. First the categorical features are encoded as binary variables using pandas “get_dummies” method.

Oversampling Fraudulent Cases Using SMOTE

Since the current dataset is highly imbalanced, in order to get equal weight on both legitimate and fraudulent transactions, we need to use a method that processes the data to have an approximate 50-50 ratio. One way to achieve this is by over-sampling, which is adding copies of the under-represented class to the data. Over-sampling is performed here using Synthetic Minority Over-sampling Technique (SMOTE). In this process, the synthetic data points are created using the k nearest neighbours.

Other over-sampling methods include using resample where observations from the minority class are randomly duplicated in order to reinforce its signal.

In resampling procedures, first the data is split into train and test sections and resampling is only done on the train dataset.

Accuracy, Precision and Recall

Accuracy, Precision and Recall are defined as follows:

- Accuracy = (TP+TN)/total
- Precision = TP/(TP+FP)
- Recall = TP/(TP+FN)

Here we are mostly interested in recall scores, to make sure we do not have many false negatives, and therefore we have captured most of the fraudulent cases. Trying to increase recall, leads to a decrease of precision. However, in our case, if we wrongly predict a fraudulent transaction and turns out not to be, is not a serious problem compared to the opposite.

Random Forest Classifier

For the Random Forest Classifier, we use GridSearchCV with cv=5 in order to get the best number of estimator. The best value for n_estimators is 100 among three considered values of 20, 50 and 100. Using this value and after fitting the model on the training data and predicting it on the test data, we can calculated precision and recall scores and the conduction matrix as below:

Precision score: 0.968988549618

```
Recall score:  0.995588235294
[[690498     65]
 [     9   2031]]
             precision    recall  f1-score   support

          0       1.00      1.00      1.00    690563
          1       0.97      1.00      0.98      2040

avg / total       1.00      1.00      1.00    692603
```

So, the model is offering 100% recall and 97% precision accuracy on the generalized unseen data (original test set) for the fraudulent case. This is a good accuracy to achieve. The above confusion matrix shows that only 9 samples were actually fraudulent but predicted as legitimate. On the other hand, 65 legitimate samples were wrongly predicted as fraudulent.

Most important features are also determined for this classifier using "feature_importances_" attribute:

	feature_importances_
errorBalanceOrig	0.458438
oldBalanceOrig	0.182455
newBalanceOrig	0.114431
newBalanceDest	0.100220
oldBalanceDest	0.039651
step	0.034901
amount	0.031889
errorBalanceDest	0.023274
type_TRANSFER	0.007971
type_CASH_OUT	0.006769

Support Vector Machine (SVM) Classifier

Before using this model, we scale the data using StandardScaler. The the LinearSVC model is run with different values for C, using GridSearchCV. C=10 is achieved as the best parameter. Using this parameter, the confusion matrix and scores are calculated as follows:

```
Precision score:  0.0265360512857
Recall score:  0.902941176471
[[622990  67573]
 [   198   1842]]
          precision    recall  f1-score   support

        0       1.00      0.90      0.95    690563
        1       0.03      0.90      0.05      2040

avg / total       1.00      0.90      0.95    692603
```

While SVM model achieves 90% recall scores for both legitimate and fraudulent cases, its precision score for the fraudulent cases is very low and it is only 3%. This means than there are many cases (67573) which are actually legitimate cases but are predicted as fraudulent. As mentioned before, this is not as important as if the fraudulent cases are missed and was classified as legitimate.

Logistic Regression Classifier

Different values for the C parameter is tested in order to get the best recall score. The best value for C parameter is found to be 1e-08 which leads to the following scores:

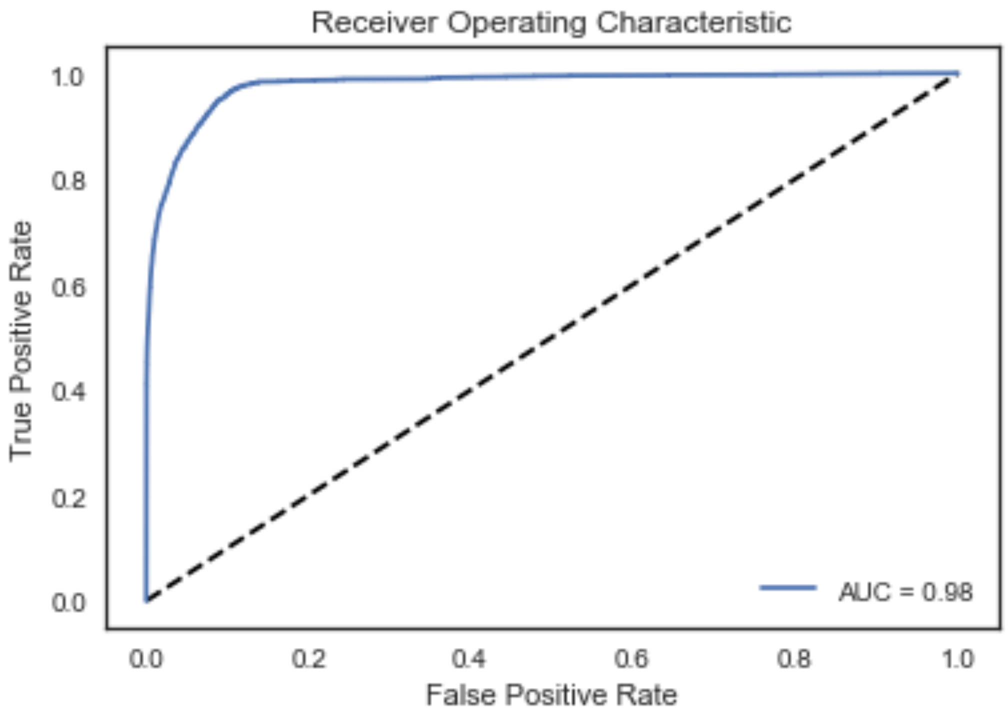
```
Cross Validation Score:  [ 0.93025195  0.92973062  0.93037746  0.93045452
0.93074536]
Logistic Regression Score:  0.890362877435
Precison score:  0.0254562735201
Recall score:  0.971568627451
```

```
[[614686  75877]
 [    58   1982]]
             precision    recall  f1-score   support

          0       1.00      0.89      0.94    690563
          1       0.03      0.97      0.05      2040

avg / total       1.00      0.89      0.94    692603
```

Receiver Operating Characteristic (ROC) Curve



ROC curves are a nice way to visualize the performance of a binary classifier, by showing how any predictive model can distinguish between the true positives and negatives. The ROC curve does this by plotting *sensitivity*, the probability of predicting a real positive will be a positive, against *specificity*, the probability of predicting a real negative will be a positive. The best decision rule is high on sensitivity and low on specificity. It’s a rule that predicts most true positives will be a positive and few true negatives will be a positive. The further the curve is from the diagonal line, the better the model is at discriminating between positives and negatives in general. There are useful statistics that can be calculated from this curve, like the Area Under the Curve (AUC). This quantifies how well the model predicts. Here the AUC is 0.98 which indicates our models is performing well.

Logistic Regression Classifier (skewed data using class_weight)

In this section, we tried logistic regression on the original data without up-sampling. Instead, the following class-weight for fraudulent and legitimate cases is calculated used for the imbalanced data.

```
[0.50148989 168.29791025]
```

In this case, the best C parameter is 0.1 and scores and confusion matrix are as follows:

```
Logistic Regression Score:  0.9978169312
Precision score:  0.996240601504
Recall score:  0.259803921569
[[690561      2]
 [  1510    530]]
             precision    recall  f1-score   support

          0       1.00      1.00      1.00    690563
          1       1.00      0.26      0.41      2040

avg / total       1.00      1.00      1.00    692603
```

As can be seen from above, using the upsampled data and class_weight leads to very good precision scores but decreases the recall score to 26% for the fraudulent cases.

Summary and Conclusions

The Paysim synthetic dataset of mobile money transactions has been analyzed. We examined the data to find features that can be discarded and the ones that can be data engineered. Some new more important features were created. To deal with the large skew in the data, we over-sampled the minority class and used different machine learning algorithms to study the performance of each model. It can be concluded that for the current data, Random Forest classifier performs the best. While both SVM and Logistic Regression result in good recall scores, the precision scores for these models are low. Using class_weight for linear regression on the original imbalanced data lead to a low recall score, which is not desired for the current case.