UNIVERSITY OF MACAU

FACULTY OF SCIENCE AND TECHNOLOGY

CISC7201

Introduction to Data Science Programming

Project Report

Financial Datasets for Fraud Detection

JIANG XINYANG (Lily)

MB955331

Contents

Introduction:

Background Information:

In this project, I choose financial dataset to analyze since I am from a business background. Financial datasets are very important for the research of fraud detection, but now there are only limited numbers of public datasets. Fortunately, I find one dataset has almost all the information I need, which can be used after some optimizations. I also add some features that I especially interested in to make the data analysis part more accurately. I think it is interesting to detect whether the transaction is fraudulent or not, so I wish to develop a methodology to make the prediction.

My Aim and Expectation:

The aim of this project is to collect, understand, clean, process and analyze data. In the analysis part, I use the method of visualization, prediction and recommendation. I try to analyze as much as I could. However, as a business student with very little background of programming, it was hard for me to finish this project perfectly and I met a lot of challenges. I try my best to make a comprehensive analysis and I am very happy that I learn a lot of new knowledge during the process of this project.

New Libraries

Except "Pandas" and "NumPy", I also use "Seaborn" and "Matplotlib" for visualization; "Statistics" for calculation and "Scikit-Learn" for changing data type and prediction. I also use the new tools like "Pydotplot" and "Graphviz" to make and export the visualization part of the random forest.

Data Collection:

Data Loading:

The data has been downloaded from web in advance, stored in "payment_data.csv." Firstly, I write code to import data and rename the original column-headers for consistency because I found some typos in the original column names. I also make the data description as comma separated values file, and then to read directly, which can save some time of loading.

Data Description:

To save time, the "description" is pre-prepared; it shows the description of all the numeric data to have an overview. The dataset that I used is downloaded from Kaggle as a public synthetic dataset generated by simulating a sample of real transactions in an African country. It is in the structure of DataFrame and it has more than 6 million records (13 columns). Its memory usage is more than 534 Megabytes.

Data Understandings:

General Understandings:

There are five types of transaction: “PAYMENT”, “TRANSFER”, “CASH_OUT”, “DEBIT” and “CASH_IN”. Among them, “Cash-Out” happens most often, as 35.17% of total; “debit” is the least frequently happening type, which only takes 0.65% of all. And if we distinguish them with normal and fraud, the percentage of fraud happening is very low, as only 0.13% of all the records. About the amount of money, the transfer type has both the maximum value 92,445,516.64 and the minimum value 0.0. And about the time of transaction taking, it turns out that every type of transactions has the similar time range.

Focus on Fraudulent Transactions:

Fraud only appears in two types of transactions: “TRANSFER” and “CASH_OUT”, with corresponding fraudulent transaction number is 4097 and 4116. We can see the two numbers are almost equal, which proves the statement provided on Kaggle: “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 an account and then cashing out of the system.” I also find that every account that has a “Fraud” record never has a second time used.

Findings and Getting Ready for the Data Cleaning:

※ ***"isFlaggedFraud" is not set accurately as the description states.***

According to the definition from Kaggle, “isFlaggedFraud” happen in “TRANSFER” and should be set when the “amount” is greater than 200,000. But I find that range of setting "isFlaggedFraud" is not the same as the description states because "isFlaggedFraud" may still not set although this condition has fulfilled. (The maximum not set "isFlaggedFraud" record is larger than 200,000.) The inaccurate rate is high as 76.77% (there are 409094 out of 532893 data that are not labelled as "isFlaggedFraud"), which is very clear that this condition is not work.

※ ***Try to find determinants about “isFlaggedFraud”.***

Since “isFlaggedFraud” is not accurately set, I wish to find out if there is any determinant about setting “isFlaggedFraud”. And I find that there are only 16 records have been flagged.

☆ ***“Step” cannot be the determinant.***

It is clear that “isFlaggedFraud” has no relationship with the number of “step” (time) because every transaction must have a “step” (no matter it is the fraudulent one or not)

☆ ***“oldBalanceOrig” and “newBalanceOrig” cannot be the determinants.***

I use range comparison in this case. After reading all the 16 flagged records, I find out that all their "oldBalanceOrig" equals to "newBalanceOrig" (originator account). I think the reason maybe is the transactions are halted. So, I check both the range of flagged records and the unflagged records with "oldBalanceOrig" equals to "newBalanceOrig" at the same time. It turns out that these two ranges have overlap, so they are not the determinants.

☆ ***“oldBalanceDest” and “newBalanceDest” cannot be the determinants.***

It is a truth that all the 16 “isFlaggedFraud” records have their "oldBalanceDest" == "newBalanceDest" == 0. The halt of transactions may also be the reason for that phenomenon. However, it shows after analysis that "isFlaggedFraud" may not be set even though both old and new “BalanceDest” is zero, so the these two are not the determinants.

☆ ***The number of transactions cannot be the determinant.***

We can find out that duplicate names are not exist within “isFlaggedFraud” transactions, but it does happen in other transactions that are not flagged as fraud. It means that “isFlaggedFraud” originators account have only started transaction once. It is clearly that “isFlaggedFraud” is independently, which means that we don’t have to consider whether the destination account has been used before or not.

☆ ***Conclusion of this part.***

Every record with “isFlaggedFraud” set has the label of “isFraud”, I think that is also part of the rule, but they did not state it in the data description. However, I think “isFlaggedFraud” seems unrelated to any explanatory variable or feature in the data. In my opinion, it is kind of useless because it is set base on the rule in the description with errors. As a result, I think that is a noisy record for analysis, so later I will drop this column from dataset.

※ <u>***“nameOrig” and “nameDest” is not useful for my analysis as well.***</u>

☆ ***Merchants not involve in the fraud.***

As I stated before, fraud only happen in “TRANSFER” and “CASH-OUT”. However, merchants only appear in the destination account of “PAYMENT”, which is very strange, and it is not the same as the definition given by Kaggle. As a result, the merchants are not involved in fraud part, which means that it is meaningless for my analysis.

☆ ***There is no common account between fraudulent "TRANSFER" and "CASH-OUT".***

From the data description, we can know that the fraudulent transactions involved in this dataset should be made by firstly make a "TRANSFER" to an account, and then use that account to make "CASH_OUT". In my point of view, the fraudulent account would be both the destination accounts of "TRANSFER" and the originator accounts of "CASH_OUT". However, after my analysis, there is no common accounts among them. As a result, I can know that the fraud are not indicated by "nameOrig" the "nameDest" because there is no chance for them to do the second use of the account since the account is set "Fraud".

※ ***<u>One reason for the data missing is because of transactions halted.</u>***

This part I only focus on the fraudulent data, which means I only focus on the "TRANSFER" and "CASH-OUT". Firstly, I do some simple data cleaning, which I will declare later in the data cleaning part. Then, I find that some transactions have the zero balance in the originator accounts, some of them have zero balance in destination accounts. But, the amount of money of these transactions is all non-zero, which cause an error-balance situation. I think maybe because of As a result, I want to calculate the missing percentage to prove my hypothesis.

☆ ***Analysis for originator accounts.***

In this case, the missing data percentage of fraudulent transactions is about 0.3%, but it is so high as 47% in the normal transactions. It may because of other errors, but it is not the same as my expectation because I think the data missing in fraudulent transactions should be more than the normal transactions. As a result, I keep finding another situation, the zero balance of the destination accounts.

☆ ***Analysis for destination accounts.***

In this case, the percentage that zero likely denotes a missing value is much larger in fraudulent transactions (49%) comparing to normal transactions (0.06%). But we cannot do anything to the missing value (0) because that is a way of estimate whether the transaction is fraudulent or not.

☆ ***Conclusion for this part.***

I know that there are many data with missing value (0) for the fraudulent transactions because that is a way of distinguishing the fraud. However, there are some other missing causing by other error, but I cannot answer this part because of my knowledge limited. I should do more research in the future to find it out. (I think both of these two missing have meaning.)

Data Cleaning:

Check Missing Values:

I check if there is any not available data. I make a plan of how to fill those NaN Values.

I want to use “SimpleImputer” to fill the NA value. Because the amount of money distributes extremely uneven, I plan to use “median” strategy to fill numerical missing values; and I will use “most_frequent” strategy to fill categorical missing values.

Check and Drop the “illegal data”:

I define the “illegal data”, which means those records with “balance < 0”, and I plan to drop those rows that contain “illegal” part.

Build a group “focus”:

Firstly, I make a copy of all the payment, and I select records only focus on “TRANSFER” and “CASH-OUT” and put them into the group named “focus”. And I also do the changing data type, drop useless columns and add new features “errorBalanceOrig” and “errorBalanceDest”, which I will explain in detail later.

Change data type:

“Type” is a categorical feature; it needs to be transformed into numerical values first if I want to do some calculation in the later analysis. To make it easier, I use “LabelEncoder” in “Scikit-Learn” to do this.

Drop useless columns in this analysis:

I drop the “nameOrig”, “nameDest” and “isFlaggedFraud” from the dataset because they are not useful for my analysis. The dimensionality of the dataset can also be reduced by doing this.

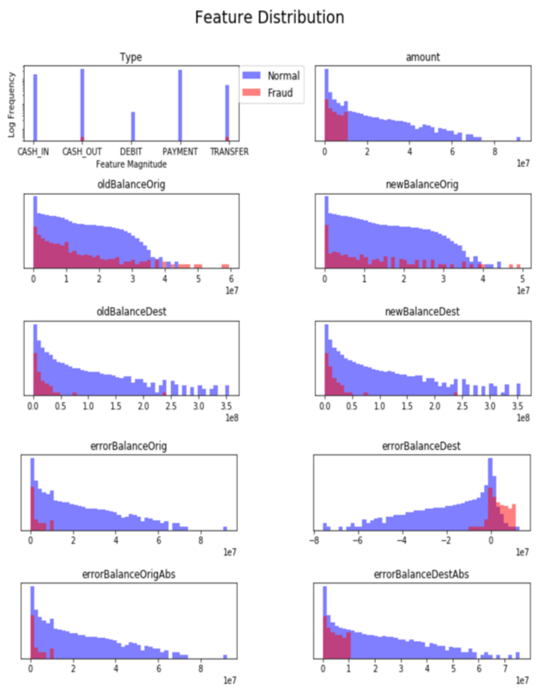
Add new features:

The amount of money is not always equaling to balance difference. So first I calculate the “balance_difference” between the old and new balance for both the originator accounts and the destination accounts. And then I calculate the difference between the amount records with the “balance_difference” and named them “errorBalanceOrig”, “errorBalanceDest” and their corresponding absolute value “errorBalanceOrigAbs” and “errorBalanceDestAbs” and then add them to the dataset.

Data Visualization:

Visualization through the whole dataset:

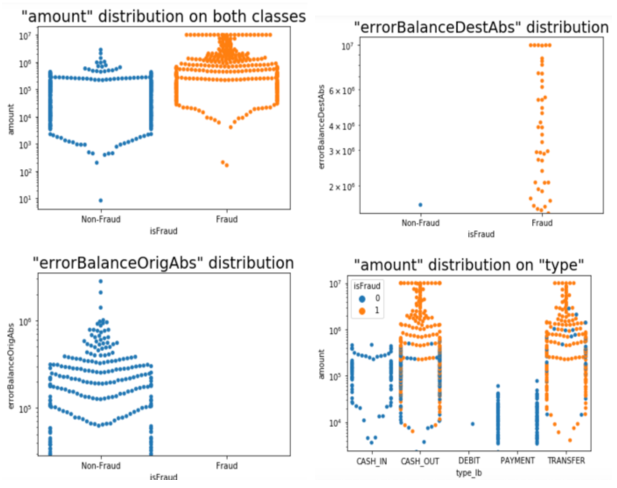
This is the Bar Graph and Swarm Plot Graph I made to show the whole picture about the data through the whole dataset. I select random sample to draw Swarm Plot to have a better look.



Bar Graph:

From the bar chart, it proves again that the fraudulent transactions only happen in "TRANSFER" and "CASH_OUT"; the number of fraudulent transactions is smaller than the normal transactions.

I am surprised that the feature "errorBalanceDest" has some visually difference between the normal and fraudulent transactions, so I keep exploring this feature.



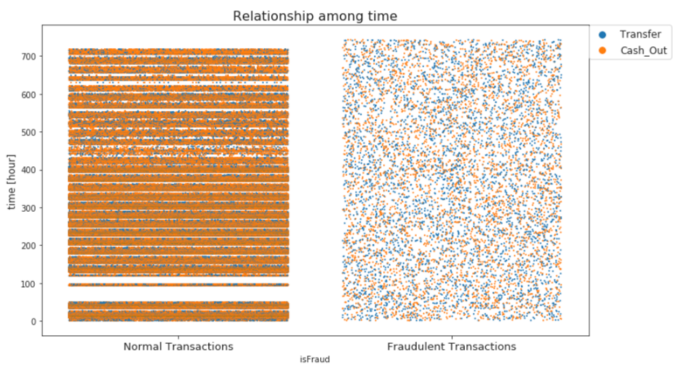
Swarm Plot Graph:

From the swarm plot, I can know that the maximum value of fraud transactions is larger than normal transactions. The number of "CASH_OUT" and "TRANSFER" transactions is larger than other types, with larger amount of money as well.

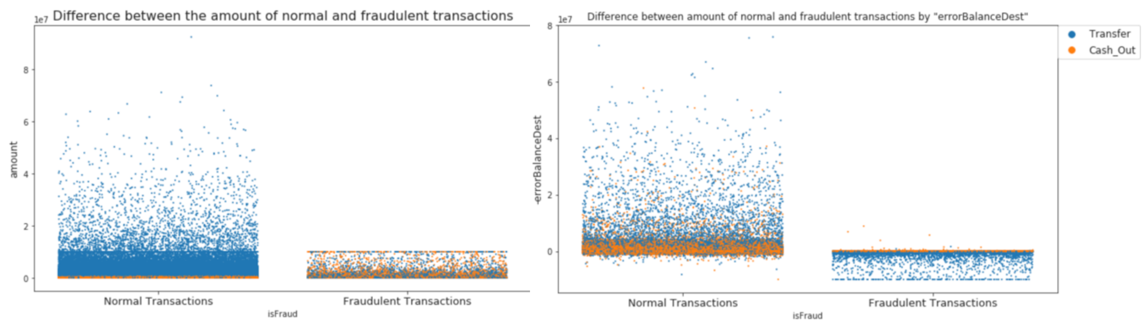
Fraud is more likely to happen when amount is unequal to the difference between old and new “BalanceDest”. (Receviers) When amount is unequal to the difference between old and new “BalanceOrig” (starters), it is more likely a normal transaction.) After calculation, the corresponding records of unequal amount with ∆BalanceOrig, ∆BalanceOrig and both are 5066425, 4122111, 3121927.

Visualization through the focus dataset:

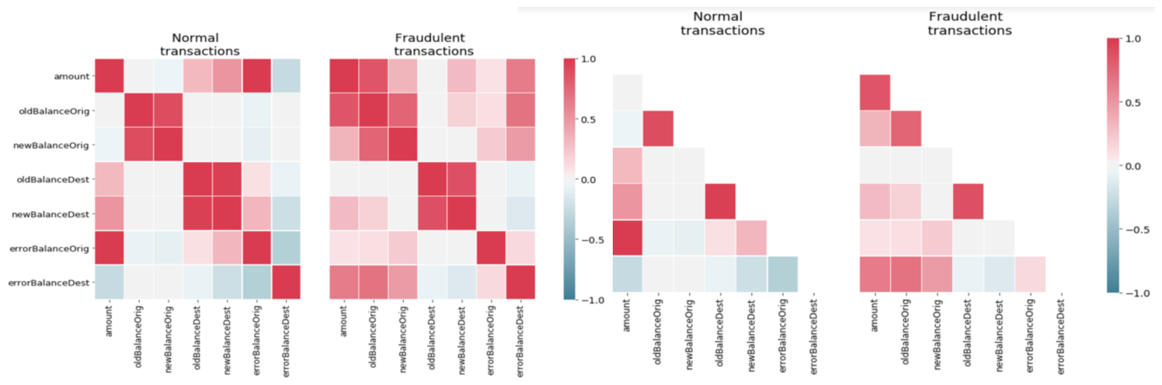
I also do the Strip Plot Visualization for the focus part to see a more clearly relationship between the fraudulent transactions and the normal transactions with the sort of transfer and cash-out.



This graph is about the relationship among time with fraud and normal transactions. The time of fraud distribute more evenly than the normal. In the normal transactions, it can be found that its number of Cash-Out is much larger than the number of transfers, and the cash-out time is longer than the transfer time.



The graph on the left side shows that the amount of normal transactions is much larger than the fraudulent transactions. And the graph on the right side shows the relationship among the newly created feature "errorBalanceDest", and I can see there is a very clear different distribution between these two groups. As a result, I think "errorBalanceDest" is a good category to compare and distinguish the fraudulent transactions.



These two graphs show the correlation between categories. (The right side one is with the mask to hide the symmetric part) And I can have a comparison between fraud and normal. It turns out that the newly added "errorBalanceOrig" and "errorBalanceDest" are very useful because these two have the biggest differences between fraud and normal transactions.

Recommendation

① Now the number of public datasets is too limited because the company is not willing to public their dataset, but it is not convenient for those data analyses, especially for those analyses aim to detect the financial fraud. As a result, I think some policy should be used to encourage the company to "public" their dataset to a certain extent (give out data to the specific group of people to analyze and prediction), which can benefit both the company and the public for increasing the safety of financial transactions. Because every country or even every city has its own situation, it is very hard for us to only use one dataset to represent all the financial transactions over the world. Even though the prediction can help some people, the worldwide financial problem cannot be improved a lot.

② After the fraud happened, the detection system can work and find it out. However, it cannot help to prevent or decrease the number of frauds happening. As a result, I think the organization, whatever bank or financial company, should make a database of the fraud information with not only the account name (because fraud maker could have many different accounts), but also the personal information. The personal information should be controlled only by the inner company, but it can be saved as a record. When some other people trying to make transfer to that person, the system may have an alert to the customer that the destination account owner has the history of fraud, which will make the customer think about whether to do the transfer or not. In that case, this fraudulent transaction may not take place. I think we should find out ways to really cut down the number of fraudulent transactions.

Prediction by using Random Forest

Model Selection

I will use different combinations of features, to train my model and test performance. At first, I will compare different classification method.

Adjusting Data

In the original dataset, the number of non-Fraud cases is way larger than Fraud cases. If I use keep this class ratio to train our model, it will always predict new data to non-Fraud cases. So, I have to find a proper way to use our data. To make things easier, I simply pick the same number of nonfraud cases as fraud ones, to train our model. And I will use the full dataset later.

Comparison of Decision Tree and Logistic Regression

I use only numerical features to make this comparison. After comparing, I find out that the Decision Tree model has a higher accuracy than Logistic Regression in this dataset.

Feature Selection

At first, I use all the numeric features, using the Decision Tree model to see the accuracy. And then, all the features except those have already dropped is used. It turns out that the Decision Tree model achieves a high accuracy with all the features used.

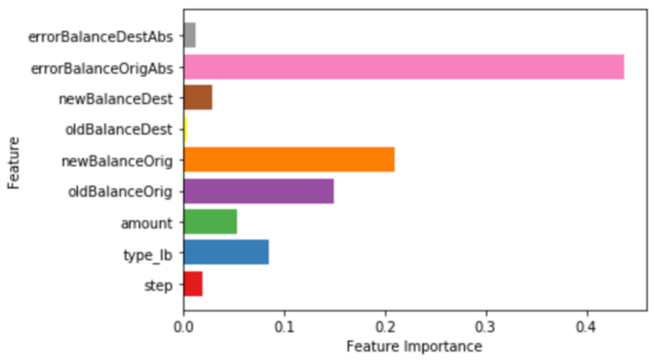
Improving Model

☆ ***Pre-pruning***

I did not give any constraint on my Decision Tree model, so it will go as deep as possible, until train set accuracy is 100% (if possible). But this is a symbol of overfitting. In order to improve generalization performance, I can do some pruning to prevent it from overfitting.

☆ ***Features importance***

The graph on the right shows that the features "errorBalanceOrigAbs" and "errorBalanceDestAbs" that I just created are important and have most relevant for the model.



☆ ***Random Forest Classifier***

Single Decision Tree is limited in many aspects. Using many decision trees combined can reduce overfitting and give better generalization performance. This is also called a Random Forest.

Random Forest Classifier on Full Dataset

In previous process we only use 1/1000 data, and now more of them will be used. I will build 1000 decision trees using different data sampled from full dataset.

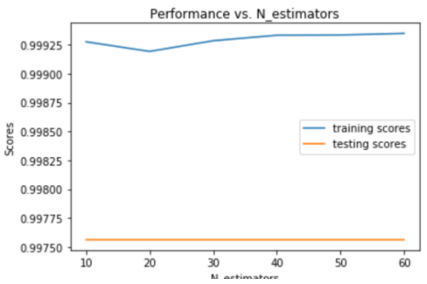
At first, I make "Self-defined random forest classifier", and then making "Train Test Split on Full Data" that taking 1/10 from fraud cases into test case, and the same amount from nonfraud cases. The rest data are used to train model.

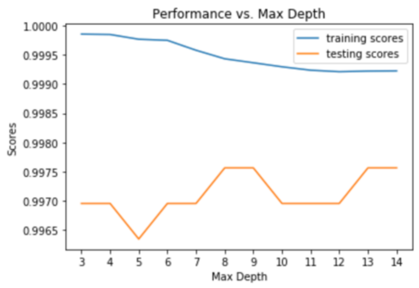
Model Training

For each tree in "MyForest", I use the full fraud train set and a same-size sample from non-fraud train set to train it.

Parameter Optimization

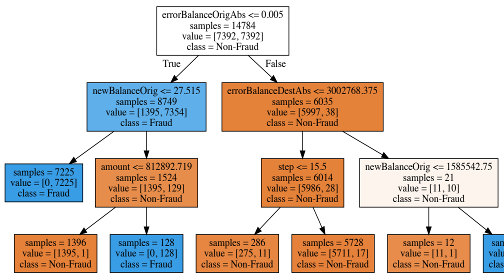
I try to find the best values for "n_estimators" and other parameters. However, this section takes long times to run, and the improvement is limited. As a result, I think it's alright to skip this section. This part includes the "Optimizating n_estimator" and "Optimizating max_depth". And the results are showing by graph.

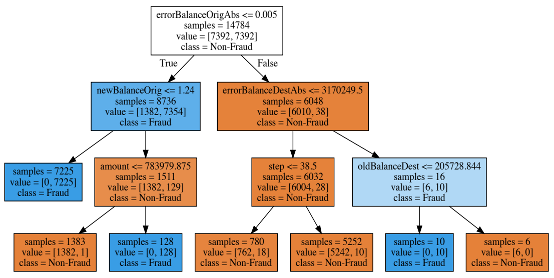


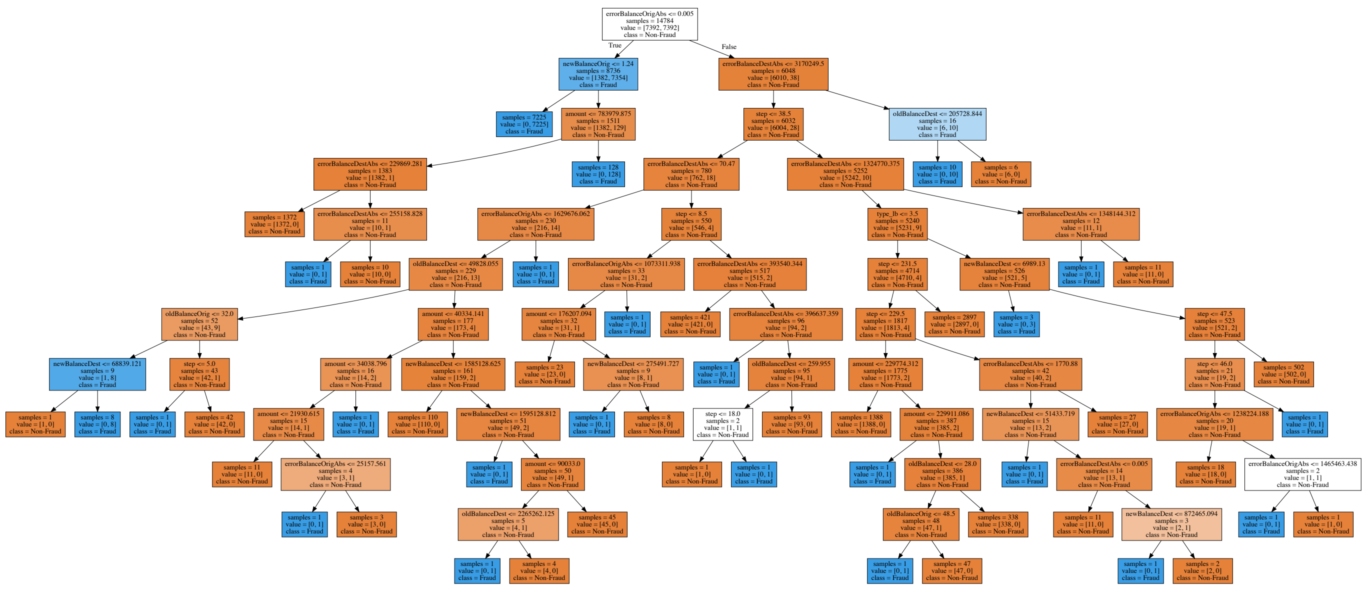


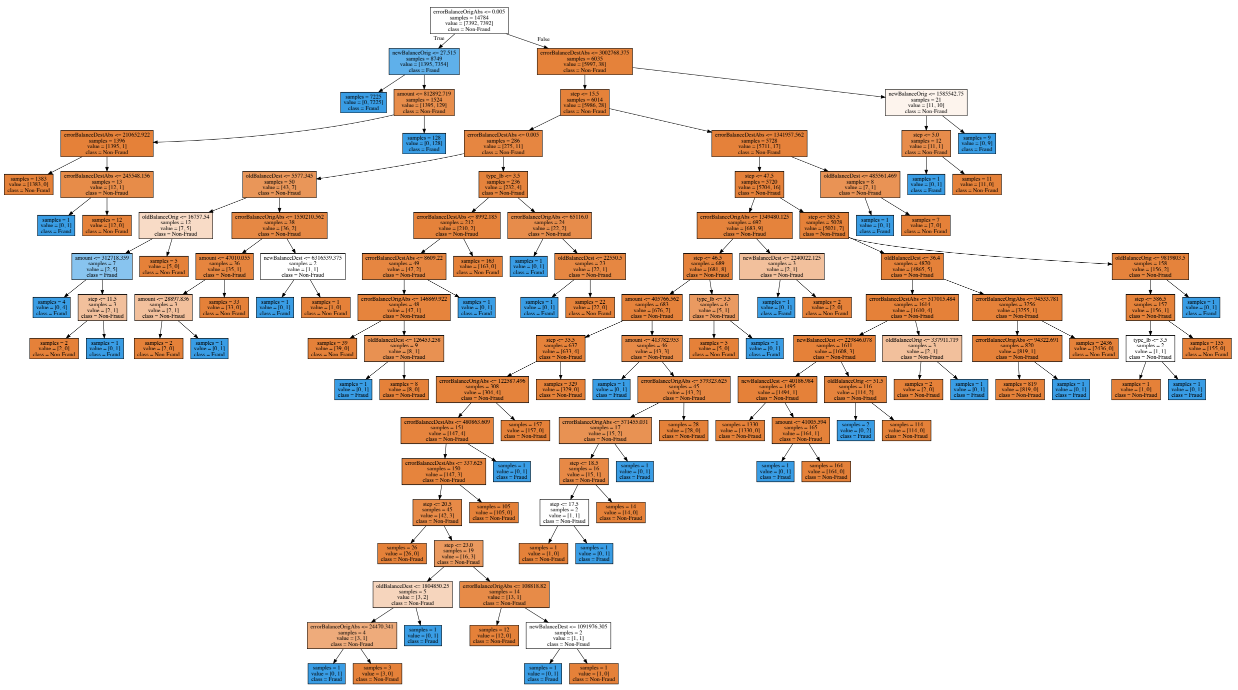
Model Visualization

In order to make the visualization more conveniently, I download the tool "Graphviz" and use the "pydotplus" to visualization the part of simple tree and the MyForest and save to pdf files. And it is shown here. (only visualization some of them)









Conclusion

From this analysis, I know the basic situation and the relationship between the fraudulent and the normal transactions. I know that the number of frauds is not so much comparing with the normal transactions, but the amount of money is usually high. The newly created feature "errorBalanceDest" and "errorBalanceOrig" are very useful for this analysis. "errorBalanceOrig" has a close relationship with the machine learning model; and the "errorBalanceDest" is one of the useful categories to distinguish the fraudulent and normal transactions.

I also know that Fraud is more likely to happen when amount is unequal to the difference between old and new BalanceDest (Receviers). When the amount is not equal to the difference between old and new BalanceOrig (starters), it is more likely a normal transaction.

However, there is still something not really perfect. I think this dataset cannot represent all the situation. Now I am using this dataset to analysis and do training, although the prediction is done, we cannot think that the fraud only happens in "CASH_OUT" and "TRANSFER" every time, which means that if a fraud happen in the other type of transaction, this model may not predict it correctly. And also, there might be data missing, this problem should be solved in the future and I should think more about it.

Reference

1. E. A. Lopez-Rojas , A. Elmir, and S. Axelsson. (2016) “PaySim: A financial mobile money simulator for fraud detection”.

2. J. Ali, R. Khan, N. Ahmad, I. Masqsood. (2012) “Random Forests and Decision Trees.”

3. Saloni. GeeksforGeeks: Plotting graph using Seaborn in Python, from https://www.geeksforgeeks.org/plotting-graph-using-seaborn-python/

4. R. R. Deshmukh, V. Wangikar. (2011) “Data Cleaning: Current Approaches and Issues”.