CREDIT CARD FRAUD DETECTION PREDICTIVE MODELING

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Nishant Sharma

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

May 2019

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Credit Card Fraud Detection Predictive Modeling

By

Nishant Sharma

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall E. Nygard
Chair

Dr. Anne Denton

Dr. Ronald Degges

Approved:

9/16/2020	Dr. Kendall E. Nygard
Date	Department Chair

ABSTRACT

Finance fraud is a growing problem with consequences in the financial industry and data mining has been successfully applied to huge volume of complex financial datasets to automate and analyze credit card frauds in online transactions. Data Mining is challenging process due to two major reasons–first, profiles of normal and fraudulent behaviors change frequently and second, card fraud data sets are highly skewed.

This paper investigates and checks the performance of Random Forest Classifier, AdaBoost Classifier, XGBoost Classifier and LightGBM Classifier on highly skewed credit card fraud data. Dataset of credit card transactions is sourced from European cardholders containing 284,786 transactions. These techniques are applied on the raw and preprocessed data. The performance of the techniques is evaluated based on accuracy, sensitivity, specificity, precision. The results indicate about the optimal accuracy for Random Forest, AdaBoost, XGBoost and LightGBM classifiers are 85%, 83%, 97.4%, and 93% respectively.

ACKNOWLEDGEMENTS

I express my sincere thanks to my advisor and graduate committee's chairperson Dr. Kendall. E. Nygard for believing in me and providing me the opportunity to pursue my degree at North Dakota State university. The encouragement and guidance throughout the research project and coursework was of paramount importance. I would like to thank my committee members, Dr. Anne Denton and Dr. Ronald Degges for their valuable guidance and helpful comments.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

1. INTRODUCTION

Financial fraud is a growing concern with far reaching consequences in the government, corporate organizations, finance industry, In Today's world high dependency on internet technology has enjoyed increased credit card transactions but credit card fraud had also accelerated as online and offline transaction. As credit card transactions become a widespread mode of payment, focus has been given to recent computational methodologies to handle the credit card fraud problem. There are many fraud detection solutions and software which prevent frauds in businesses such as credit card, retail, e-commerce, insurance, and industries. Data mining technique is one notable and popular methods used in solving credit fraud detection problem. It is impossible to be sheer certain about the true intention and rightfulness behind an application or transaction. To seek out possible evidences of fraud from the available data using mathematical algorithms is the best effective option. Fraud detection in credit card is the truly the process of identifying those transactions that are fraudulent into two classes: legit and fraud class transactions, several techniques are designed and implemented to solve to credit card fraud detection such as genetic algorithm, artificial neural network, frequent item set mining, machine learning algorithms, migrating birds optimization algorithm, can also perform comparative analysis of random forest, AdaBoost, XGBoost and LightGBM. Credit card fraud detection is a very popular but also a difficult problem to solve. Firstly, due to issue of having only a limited amount of data, credit card makes it challenging to match a pattern for dataset. Secondly, there can be many entries in dataset with fraud transactions which fits the pattern of legitimate behavior. Also, the problem has many constraints. Firstly, sensitive data sets are not easily accessible for public and the results of researches are often hidden and censored, making the results inaccessible and due to this it is sometimes challenging to benchmark certain models. Secondly, the improvement of methods is more difficult by the fact that the security concern imposes a limitation to exchange of ideas and methods in fraud detection, and especially in credit card fraud detection. Lastly, the data sets are continuously evolving and changing which makes the profiles of normal and fraudulent behaviors being different from the legit transaction in the present, which may have been fraud in past or vice versa. This paper evaluates four advanced data mining approaches, Decision tree, support vector machines, Logistic regression, and random forests and then a collative comparison is made to evaluate that which model performed best.

Credit card transaction datasets are rarely available, highly imbalanced, and skewed. Optimal feature (variables) selection for the models, suitable metric is most important part of data mining to evaluate performance of techniques on skewed credit card fraud data. Fraudulent behavior profile is dynamic, that is fraudulent transactions tend to look like legitimate ones, also credit card fraud detection performance is greatly affected by type of sampling approach used, In the end of this paper, conclusions about results of classifier evaluative testing are made and collated. From the experiments the result that has been concluded is that Random Forest Classifier has a accuracy of 85% while AdaBoost shows accuracy of 83%, and LightGBM shows accuracy of 93% but the best results are obtained by XGBoost with a precise accuracy of 97%. The results obtained thus conclude that XGBoost Classifier shows the most precise and high accuracy of 97% in problem of credit card fraud detection with dataset provided by ULB machine learning.

2. RELATED WORK

In [9] This paper case study is performed regarding credit card fraud detection, where data normalization is applied before Cluster Analysis. Results obtained from the use of Cluster Analysis and Artificial Neural Networks on fraud detection shown that by clustering attributes neuronal inputs can be minimized. Promising results can be obtained by using normalized data and data should be MLP trained. This research is based on unsupervised learning. Significance of this paper is to find new methods for fraud detection and to increase the accuracy of results.

In [10] In this paper a new collative comparison measure that reasonably represents the gains and losses due to fraud detection is proposed. A cost sensitive method which is based on Bayes minimum risk is presented using the proposed cost measure. This method shows efficiency of 23% as compared to other state of the art algorithms. The data set for this paper is based on real life transactional data provided by a large European company as personal details in data are kept confidential., accuracy of an algorithm is around 50%. Significance of this paper is to find an algorithm that can reduce the cost measure. Bayes minimum risk algorithm is obtained by efficiency of 23% over other algorithms.

In [11] Various modern techniques based on Sequence Alignment, Machine learning, Artificial Intelligence, Genetic Programming, Data mining etc. has been evolved and is still evolving to detect fraudulent transactions in credit card. A sound and clear understanding on all these approaches is required that can certainly lead to an efficient credit card fraud detection system. Survey of various techniques used in credit card fraud detection mechanisms has been shown in this paper along with evaluation of each methodology based on certain design criteria. Analysis on Credit Card Fraud Detection Methods has been done. The survey in this paper is purely subjected to detect efficiency and transparency of each method. Significance of this paper was conducting a survey to compare different credit card fraud detection algorithm to find the most suitable algorithm to solve the problem.

In [12] A comparison has been made between models based on artificial intelligence. General descriptions of the developed fraud detection system are provided in this paper such as Naive Bayesian Classifier based on Bayesian Networks and clustering model. Number of legal truncations determined are greater or equal to 0.65 means Bayesian Network accuracy is 65% for provided data set.

In [13] Nutan and Suman provided extensive review of credit card fraud detection methods and have supported the theory of credit card fraud, types of fraud like telecommunication, bankruptcy fraud etc. and how to detect it. In addition, they have explained numerous algorithms and methods on how to detect fraud using Glass Algorithm, Bayesian, networks, Hidden Markova model, Decision Tree and 4 more. They have explained in detail about each algorithm and how this algorithm works along with mathematical explanation. Types of machine learning along with classifications are mentioned. Pros and cons of each method is listed. This research is to detect the credit card fraud in the dataset obtained from ULB by applying Logistic regression, Decision tree, SVM, Random Forest and to evaluate their Accuracy, sensitivity, specificity, precision using different models and compare and collate them to state the best possible model to solve the credit card fraud detection problem.

3. PROBLEM STATEMENT

3.1. Objectives

The problem statement clearly explained the issue of cybersecurity and how we can analyze it using modern datamining techniques. It is difficult to analyze or detect fraud transactions without help of machine learning classifiers. This kernel helps to understand data with basic visualization and statistical methods.

3.2. Dataset

In this Experiment, we used a dataset which consist of transactions made by European cardholders during month of July 2015 through various credit cards. Data has been collected and analyzed during a research collaboration with worldline and machine learning group (http://mlg.ulb.ac.be) of ULB(Université Libre de Bruxelles) on big data mining and fraud detection. The dataset consists of transactions occurred in span of two days, where we have 492 fraudulent transactions out of 284,807 total transactions. The dataset is highly unbalanced where positive class (Frauds) accounts for only 0.172% of all transactions.

Data set contains only numerical(continuous) input variable which are result of Principle Component Analysis (PCA) feature selection transformation, resulting in 28 principal components and total 30 input features are utilized in this study. Behavioral characteristic of the card is shown by a variable of each profile usage representing the spending habits of the customers along with days of the month, hours of the day, geographical locations, or type of the merchant where the transaction takes place. Due to confidentiality issues original features and more background information about data is not provided. Data provided as:

- Features V1, V2, ... V28 are the principal components obtained with PCA;
- The only features which have not been transformed with PCA are Time and Amount. Feature Time contains the seconds elapsed between each transaction and the first

transaction in the dataset. The feature Amount is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning.

- Feature Class is the response variable and it takes value 1** in case of fraud and **0 otherwise.

4. BACKGROUND

Ability of system to automatically learn and improve from experience without being explicitly programmed is called machine learning and it focuses on the development of computer programs that can access data and use it train itself. Classifier is the learning algorithm that learns the model from training data, or we can say it is a special case of hypothesis or model that assign a class label to data point. It is an instance of supervised learning i.e. where training set of correctly identified observations is available.

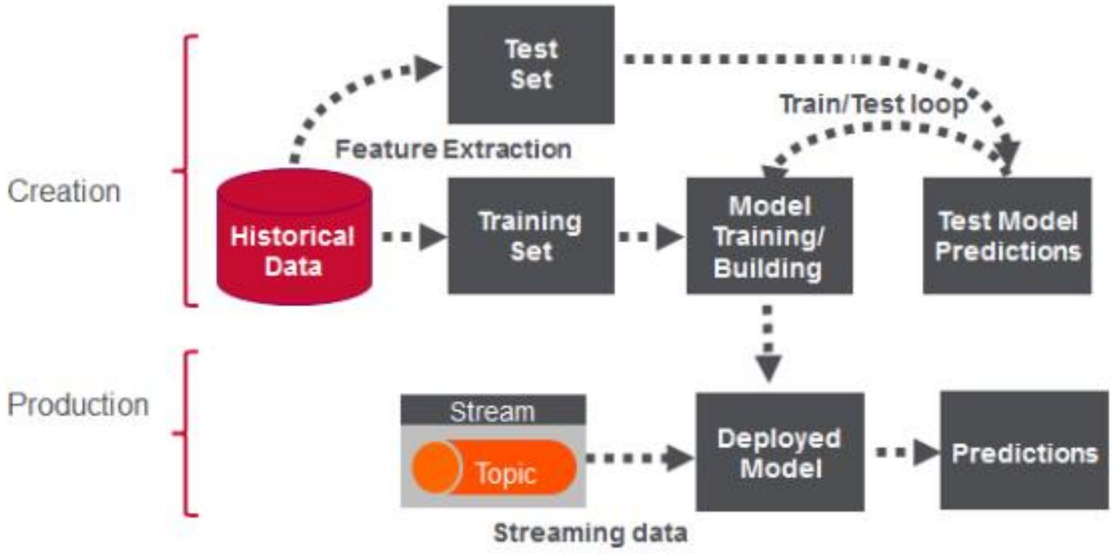
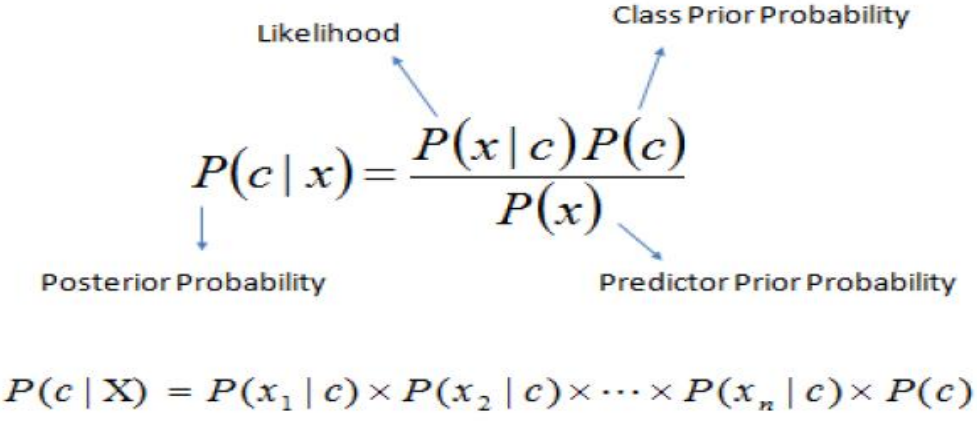


Figure 1. Classifier Steps

4.1. Random Forest Classifier

Random Forest is an algorithm for classification and regression. Summarily, it is a collection of decision tree classifiers. Random forest has advantage over decision tree as it corrects the habit of overfitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others. The Random Forest algorithm has been found to provides a good estimate of the generalization error and to be resistant to overfitting.



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

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

Figure 2. Random Forest Tree Classifier Algorithm

Random forest ranks the importance of variables in a regression or classification problem in a natural way can be done by Random Forest.

4.2. AdaBoost Classifier

Ada-boost, like Random Forest Classifier is another ensemble classifier. Ada-boost classifier combines weak classifier algorithm to form strong classifier. A single algorithm may classify the objects poorly, If we combine multiple classifiers with selection of training set at every iteration and assigning right amount of weight in final voting, we can have good accuracy score for overall classifier. It retains the algorithm iteratively by choosing the training set based on accuracy of previous training. The weightage of each trained classifier at any iteration depends on the accuracy achieved. After training a classifier at any level, ada-boost assigns weight to each training item. Misclassified item is assigned higher weight so that it appears in the training subset of next classifier with higher probability. After each classifier is trained, the weight is assigned to the classifier as well based on accuracy. More accurate classifier is assigned higher weight so that it will have more impact in outcome.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Figure 3. AdaBoost Classifier

- h_t(x) is the output of weak classifier t for input x
- alpha_t is weight assigned to classifier.
- alpha_t is calculated as follows:
- alpha_t = 0.5 * ln ((1—E)/E) : weight of classifier is straight forward, it is based on the error rate E.

  Initially, all the input training example has equal weightage.

4.3. XGBoost Classifier

XGBoost is one of the fastest implementations of gradient boosted trees. It does this by tackling one of the major inefficiencies of gradient boosted trees: considering the potential loss for all possible splits to create a new branch (especially if you consider the case where there are thousands of features, and therefore thousands of possible splits). XGBoost tackles this inefficiency by looking at the distribution of features across all data points in a leaf and using this information to reduce the search space of possible feature splits.

Although XGBoost implements a few regularization tricks, this speed up is by far the most useful feature of the library, allowing many hyperparameter settings to be investigated quickly. This is helpful because there are many, many hyperparameters to tune. Nearly all of them are designed to limit overfitting (no matter how simple your base models are, if you stick thousands of them together, they will overfit).

The list of hyperparameters was super intimidating to me when I started working with XGBoost, so I am going to discuss the 4 parameters I have found most important when training my models so far (I have tried to give a slightly more detailed explanation than the documentation for all the parameters in the appendix).

My motivation for trying to limit the number of hyperparameters is that doing any kind of grid / random search with all of the hyperparameters XGBoost allows you to tune can quickly explode the search space. I’ve found it helpful to start with the 4 below, and then dive into the others only if I still have trouble with overfitting.

4.4. LightGBM Classifier

LightGBM is a gradient boosting framework that uses tree-based learning algorithm. It grows tree vertically while other algorithm grows trees horizontally meaning that the Light GBM grows free leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

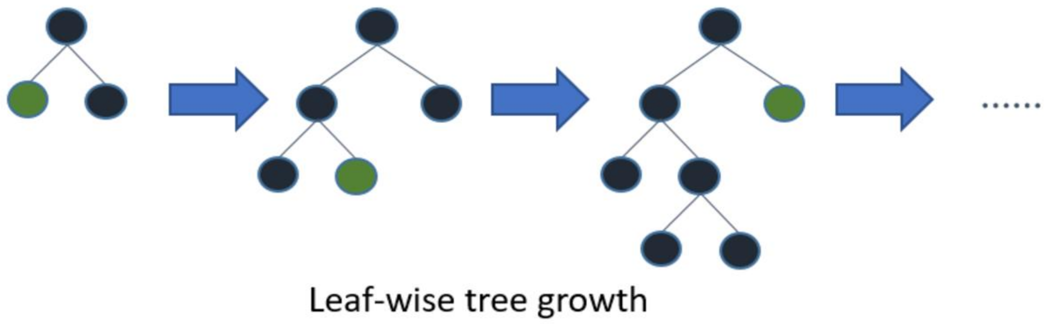


Figure 4. LightGBM Classifier

4.5. Area Under Curve

In Machine Learning, performance measurement is an essential task. So, when it comes to a classification problem, we can count on an AUC - ROC Curve. When we need to check or visualize the performance of the multi - class classification problem, we use AUC (Area Under the Curve) ROC (Receiver Operating Characteristics) curve. It is one of the most important evaluation metrics for checking any classification model’s performance.

AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represent degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

An excellent model has AUC near to the 1 which means it has good measure of separability. A poor model has AUC near to the 0 which means it has worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means model has no class separation capacity whatsoever.

5. EXPERIMENTAL APPROACH

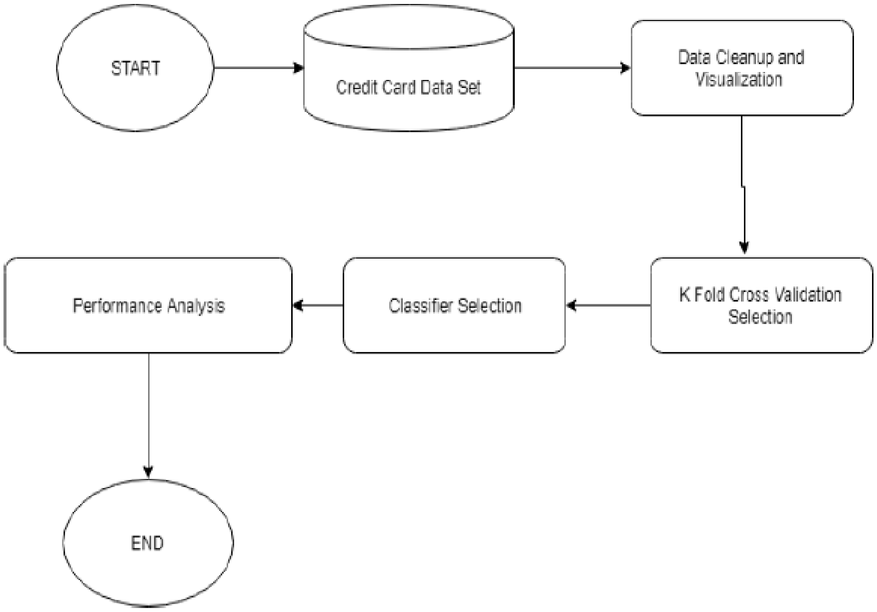


Figure 5. Architecture

Interactive process, First the credit card dataset is taken from the source and performs cleaning and validation including removal of redundancy: Process involves removal of empty columns and convert necessary variable into factors or classes. Then data is divided into 2 part, one training dataset and another test data set. Now K-fold cross validation is performed i.e. the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsample is used as training data, Models are created for Random Forest, AdaBoost, XGBoost, LightGBM and then perform data visualization, made comparison using AOC score.

5.1. Unbalanced Data

Unbalanced data refers to classification problems where we have unequal instances for different classes. Having unbalanced data is actually very common in general datasets. Mostly extreme unbalanced data is seen with fraud detection, where e.g. most credit card uses are acceptable while only very few are fraudulent. To generate bar grape, we used Plotly python graphing library using class as X axis and Y for number of transactions.

Let us check data unbalance with respect to target value, i.e. class.

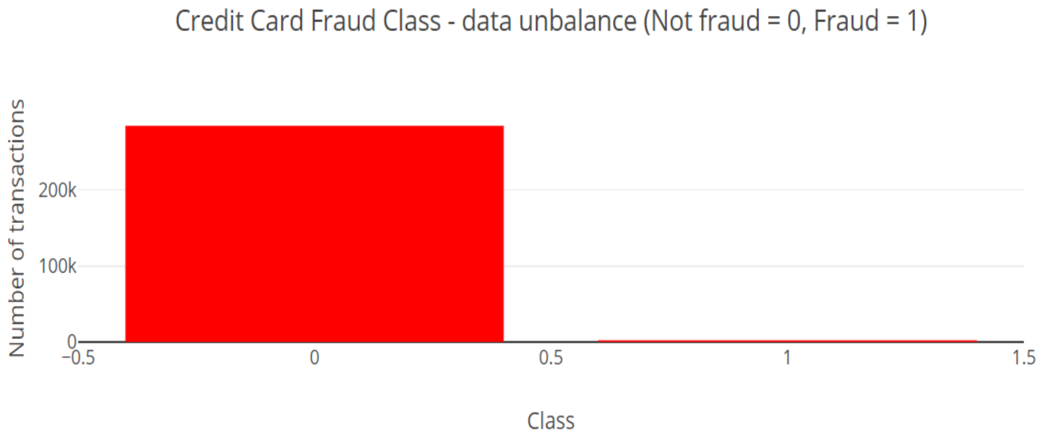


Figure 6. Fraudulent and Non-Fraudulent Transactions

Only 492 (or 0.172%) of transaction are fraudulent. That means the data is highly unbalanced with respect with target variable Class.

5.2. Exploring Data

Mentioned graph clearly shows fraudulent transactions have a distribution more even than valid transactions which are equally distributed in time, including the low real transaction times, during night in Europe time zone.

5.2.1. Transaction in Time

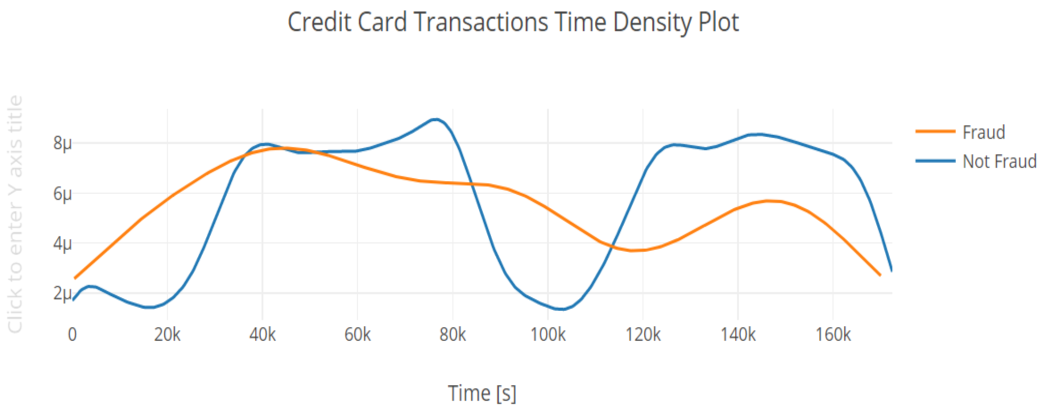
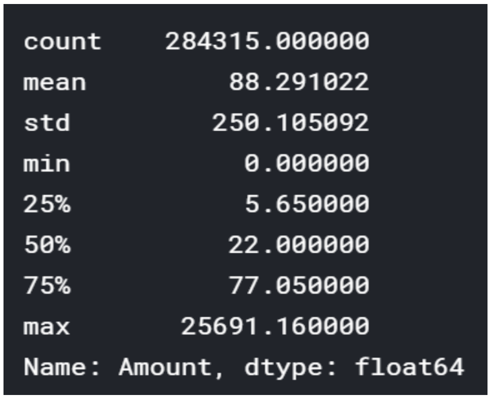
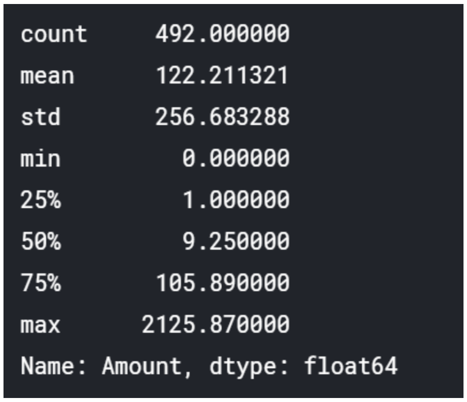


Figure 7. Transactions Time Density Plot

5.2.2. Transaction Amount



```
count     284315.000000
mean          88.291022
std          250.105092
min            0.000000
25%            5.650000
50%           22.000000
75%           77.050000
max        25691.160000
Name: Amount, dtype: float64
```



```
count      492.000000
mean       122.211321
std        256.683288
min          0.000000
25%          1.000000
50%          9.250000
75%        105.890000
max       2125.870000
Name: Amount, dtype: float64
```

Figure 8. Transaction of Non-Fraudulent and Fraudulent Amount

By Describing both classes 0 and 1 we can clearly see that the real transactions has a larger mean value, larger Q1, smaller Q3 and Q4 and larger outliers; fraudulent transactions have a smaller Q1 and mean, larger Q4 and smaller outliers.

Now we will plot the fraudulent transactions (amount) against time using Plotly graphical python package. The time is shown is seconds from the start of the period (totally 48h, over 2 days).

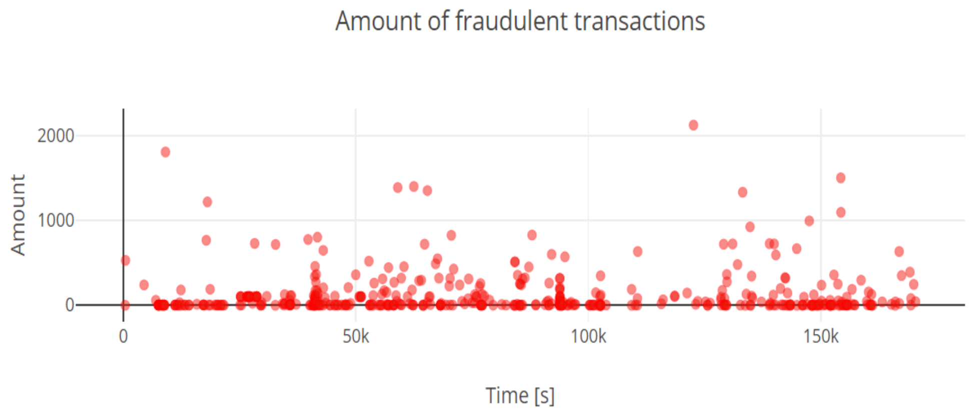


Figure 9. Fraudulent Transactions Against Time Feature Correlation

Feature correlation is a way to understand the relationship between multiple variables and attributes in our dataset. Using correlation, we can get some insight if one or multiple attributes depend on another attribute or cause of another attribute, There are 3 basic types of correlations.

- Direct Correlation: means that if feature A increases/decreases then feature B also increases/decreases. Both features move in tandem and they have a liner relationship.
- Inverse Correlation: means if feature an increase then feature B decreases and vice versa
- No Correlation: No relationship between two attributes.

Following graph shows Correlation between all 31 attributes.

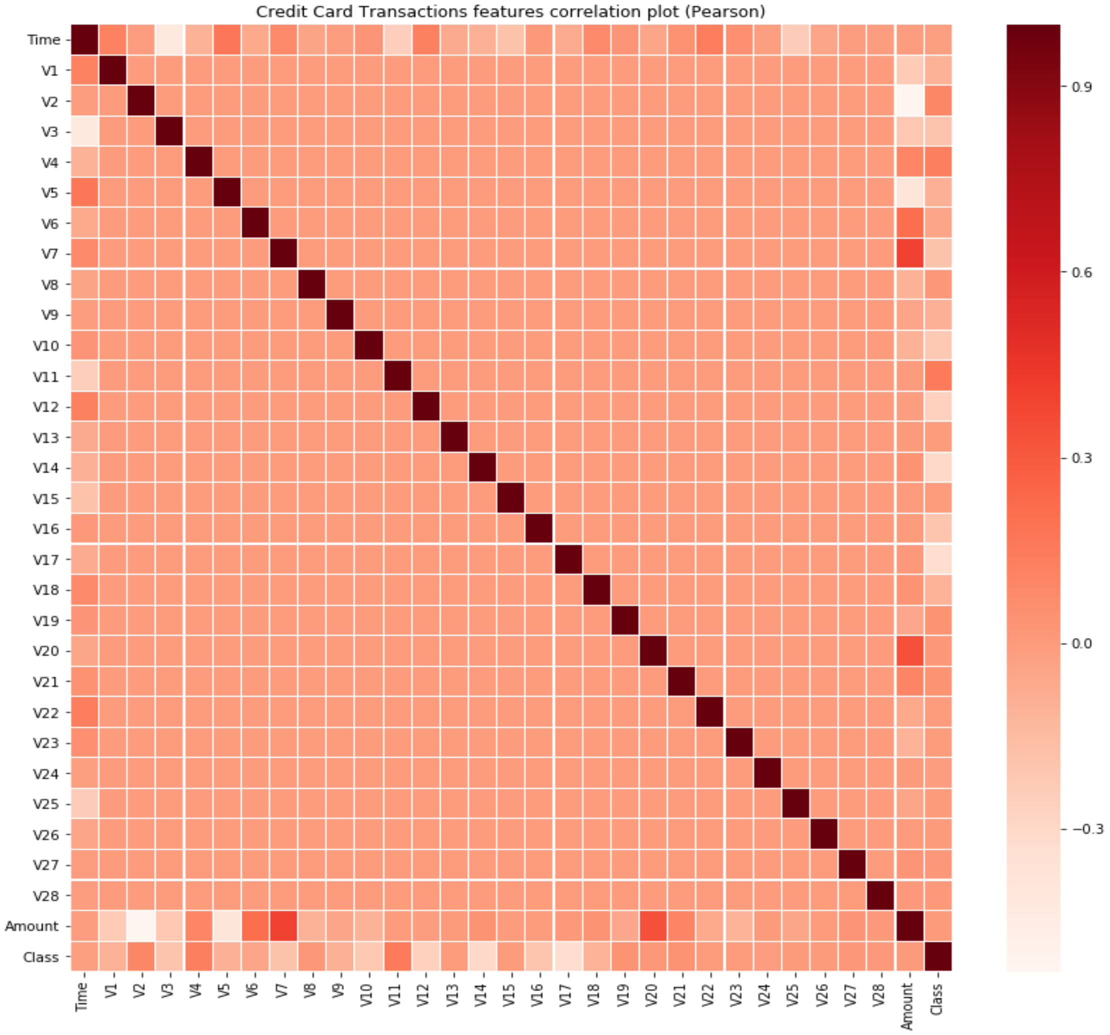


Figure 10. Feature Correlation Graph

As expected, there is no notable correlation between features V1-V28. As there are certain correlations between some of the features and Time (inverse correlation with V3) and Amount (direct correlation with V7 and V20, inverse correlation with V1 and V5). Next, we will plot the correlated and inverse correlated values on the same graph using matplotlit Python Graph packages and visualize direct correlated values: {V20; Amount} and {V7; Amount}.

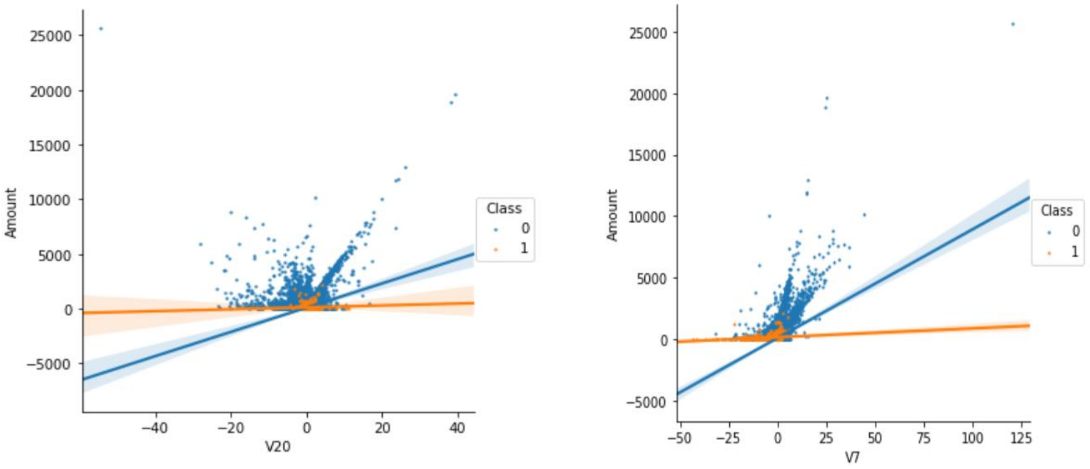


Figure 11. Direct Correlation Graph

By looking at direct correlation graph we can confirm that the two features are correlated (the regression lines for Class = 0 have a positive slope, whilst the regression line for Class = 1 have a smaller positive slope). Now we will plot the inverse correlated values using Matplotlib Python Library package.

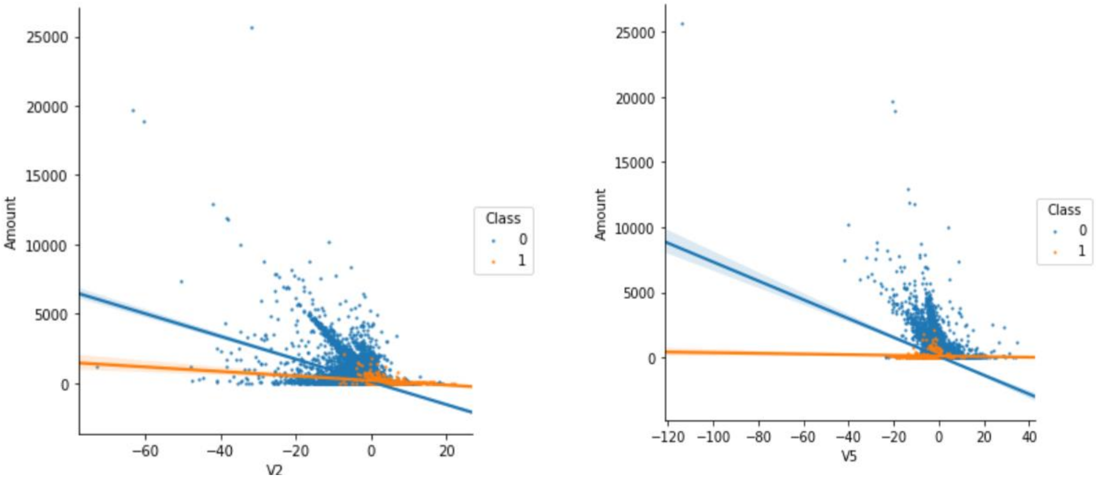


Figure 12. Inverse Correlation Graph

We can confirm that the features are inverse correlated (the regression lines for Class = 0 have a negative slope while the regression lines for Class = 1 have a very small negative slope).

5.2.3. Feature Density Plot

Feature Density Plot visualize the distribution of data over a continuous interval of time. We are using density plot which is a variation of histogram that uses kernel smoothing to plot values of various attributes. Peaks of the density plot helps to determine where values are concentrated over the interval. We are using density plot to study distribution of attribute. Checking the distribution of our attributes one by one is first thing we should do to provide good quality of information.

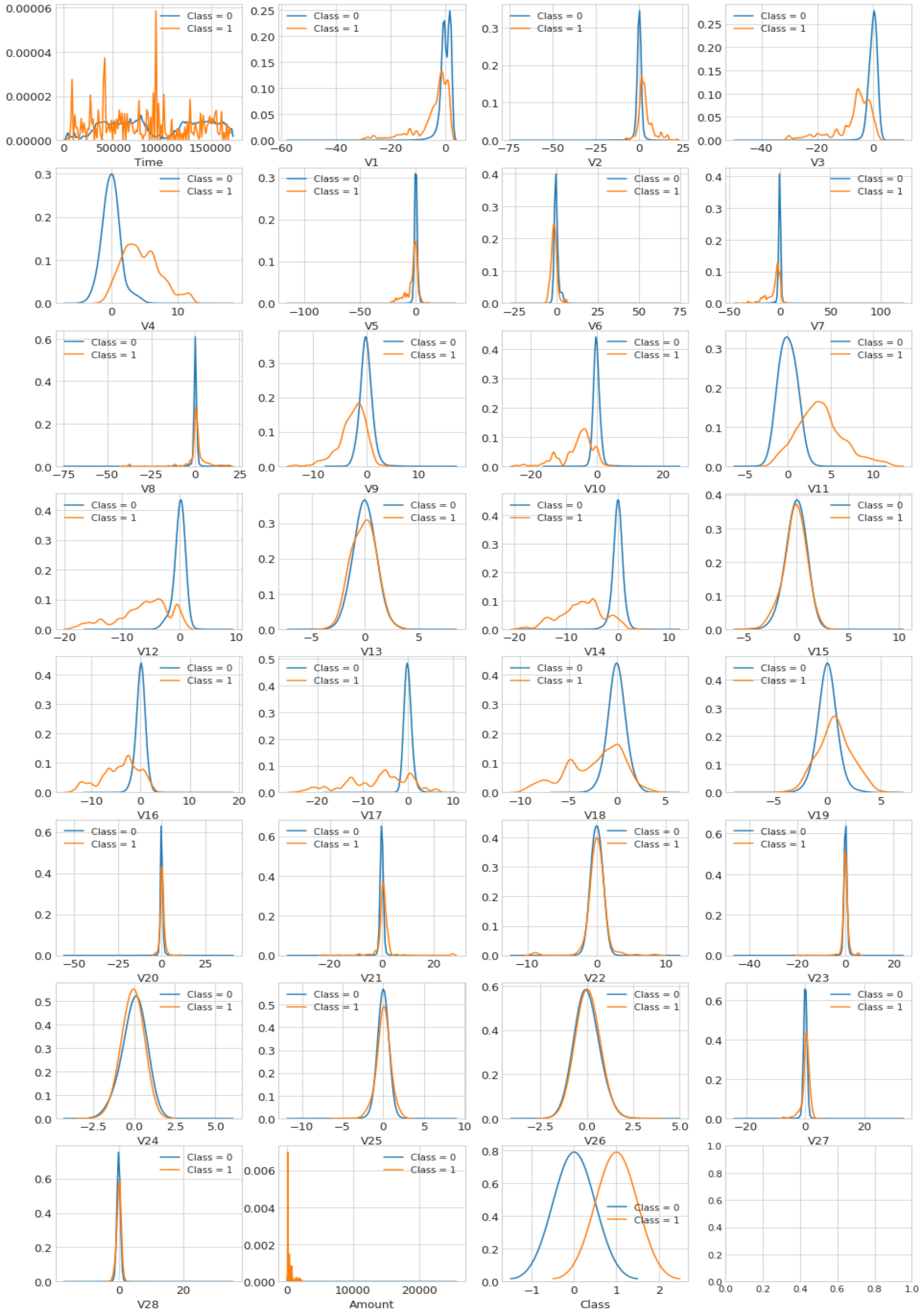


Figure 13. Feature Density Plot

For some of the features we can observe a good selectivity in terms of distribution for the two values of Class: V4, V11 have clearly separated distributions for Class values 0 and

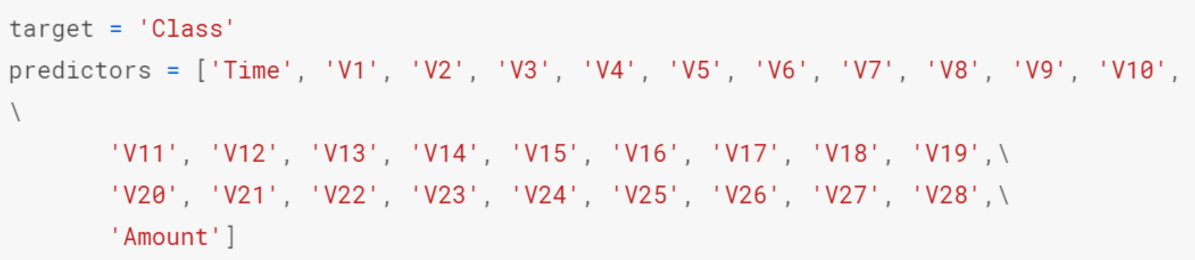
1. V12, V14, V18 are partially separated. V1, V2, V3, V10 have a quite distinct profile.

whilst V25, V26, V28 have similar profiles for the two values of Class.

In general, with just few exceptions (Time and Amount), the features distribution for legitimate transactions (values of Class = 0) is centered around 0, sometime with a long queue at one of the extremities. In the same time, the fraudulent transactions (values of Class = 1) have a skewed (asymmetric) distribution.

6. PREDICTIVE MODELING

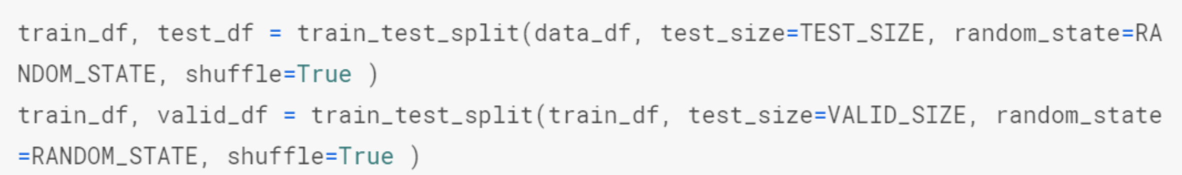
Now we will define the predictor, target, categorical features, if any. In our case, there are no categorical feature.



```
target = 'Class'
predictors = ['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
\
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19',\
       'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28',\
       'Amount']
```

Figure 14. Target and Predictor Classes

Now we will define Train, Validate and Test Data



```
train_df, test_df = train_test_split(data_df, test_size=TEST_SIZE, random_state=RA
NDOM_STATE, shuffle=True )
train_df, valid_df = train_test_split(train_df, test_size=VALID_SIZE, random_state
=RANDOM_STATE, shuffle=True )
```

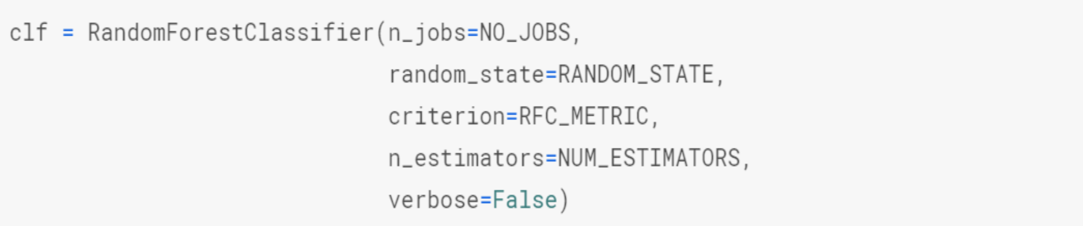
Figure 15. Train, Validate and Test Data

6.1. Random Forest Classifier

Let us run a model using the training set for training. Then, we will use the validation set for validation.

We will use as validation criterion GINI, both GINI and AUC are standard measures of accuracy for accessing the performance of credit scoring model. Both of these measures can be used in similar manner but AUC score is required in order to determine GINI, whose formula is GINI = 2 * (AUC) - 1, where AUC is the Receiver Operating Characteristic - Area Under Curve (ROC-AUC). Number of estimators is set to 100 and number of parallel jobs is set to 4.

We start by initializing the Random Forest Classifier.



```
clf = RandomForestClassifier(n_jobs=NO_JOBS,
                             random_state=RANDOM_STATE,
                             criterion=RFC_METRIC,
                             n_estimators=NUM_ESTIMATORS,
                             verbose=False)
```

Figure 16. Random Forest Classifier Model

We trained the RandonForestClassifier using the train_df data and fit function. At the same time, we will also predict the target values for the valid_df data, Using predict function. Next step we will follow is to visualize feature importance.

6.1.1. Feature Importance

Feature importance is one of the benefits of gradient boosted trees. It is straight forward process to retrieve important scores of each attribute. Generally, importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance.

This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other. To determine the feature importance of provided data we will be using Predicted target values against number of attributes.

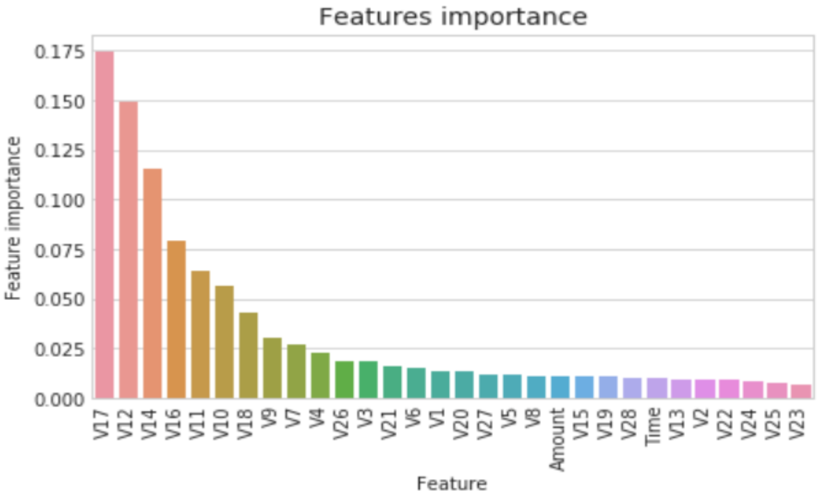


Figure 17. RFC Feature Importance

Feature importance graph can help us better understand the model's logic, we can verify it being correct but also work on improving the model by focusing only on important features. Above mentioned graph shows that most important and useful features are

V17, V12, V14, V10, V11, V16.

6.1.2. Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It also allows the visualization of the performance of an algorithm.

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

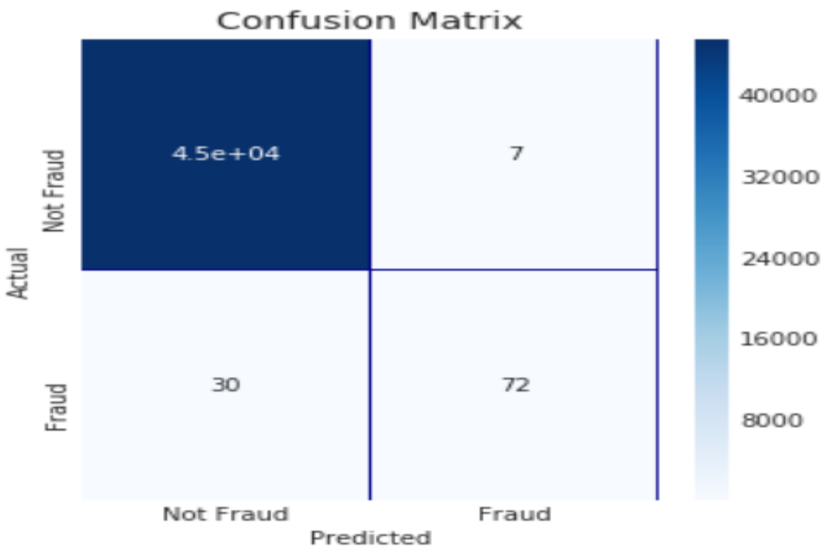


Figure 18. RFC Confusion Matrix

We will be using confusion matrix to represent different metrics that accounts for selectivity and specificity to minimize the time consumption of both Type I errors and Type II errors.

Null Hypothesis (H0) - The transaction is not a fraud.

Alternative Hypothesis (H1) - The transaction is a fraud.

- Type I error - You reject the null hypothesis when the null hypothesis is true.
- Type II error - You fail to reject the null hypothesis when the alternative hypothesis is true.
- Cost of Type I error - You erroneously presume that the transaction is a fraud, and a true transaction is rejected.
- Cost of Type II error - You erroneously presume that the transaction is not a fraud and a fraudulent transaction is accepted.

The following image explains what Type I error and Type II error:

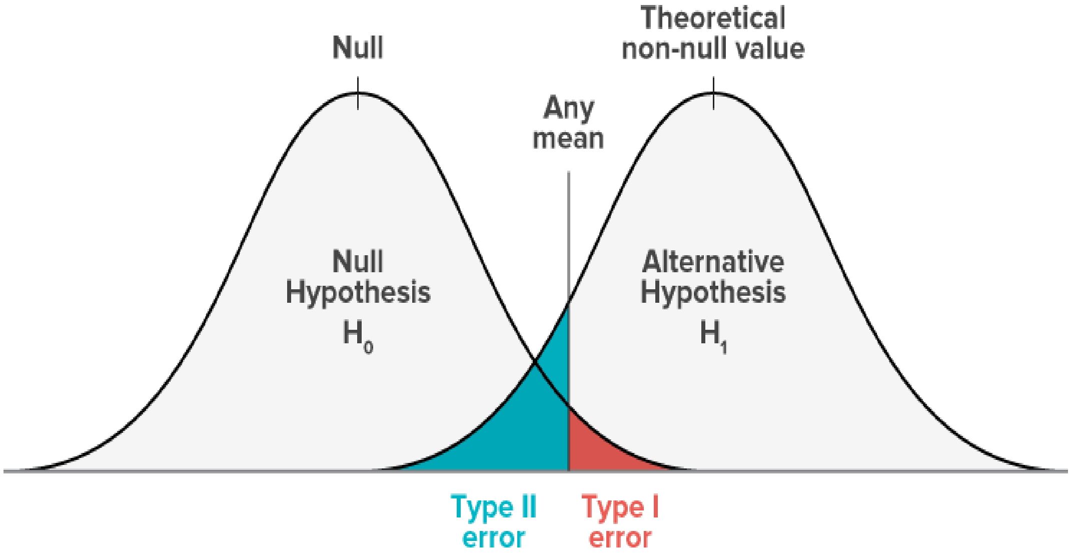


Figure 19. Type 1 and Type 2 Error

6.1.3. Calculating Area Under Curve Score

To calculate ROC – AUC score we used target values and predicted target values obtained in above steps.

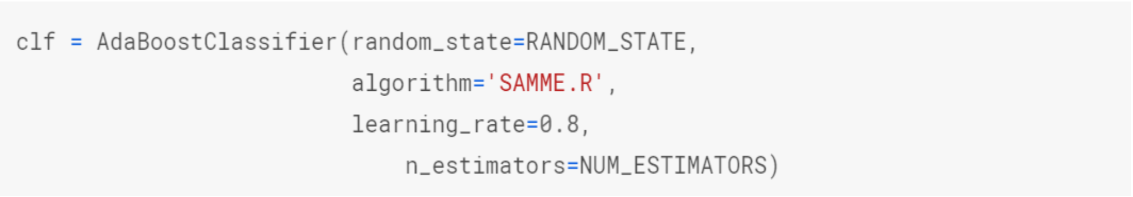
- roc_auc_score(valid_df[target]. values, preds)

The ROC-AUC score obtained with RandomForrestClassifier is 0.85.

6.2. AdaBoost Classifier

We will run adaBoost model using the training set for training. Then, we will use the validation set for validation.

We will start by initializing the Adaboost Classifier.



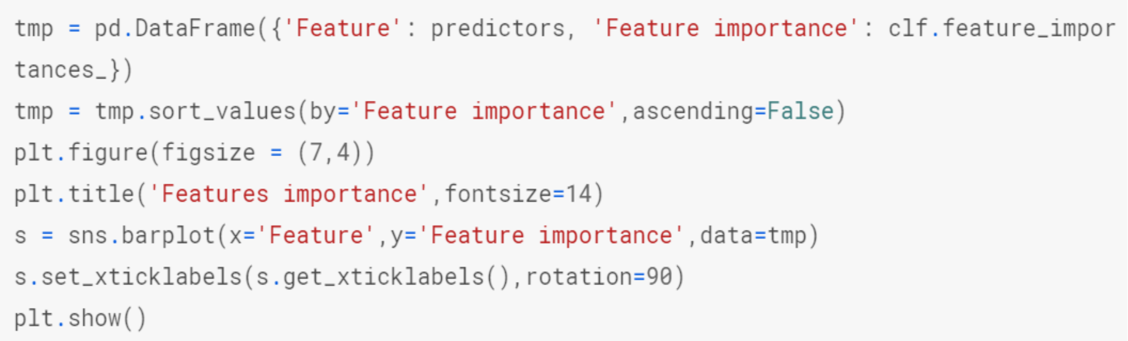
```
clf = AdaBoostClassifier(random_state=RANDOM_STATE,
                         algorithm='SAMME.R',
                         learning_rate=0.8,
                             n_estimators=NUM_ESTIMATORS)
```

Figure 20. AdaBoost Model

We trained the AdaBoost using the train_df data and fit function. At the same time, we also predicted the target values for the valid_df data, Using predict function. Next step we followed it to visualize feature importance.

6.2.1. Feature Importance

Feature importance is calculated explicitly for each attribute in the dataset which we did in the case of random forest classifier, allowing attributes to be ranked and compared to each other. To determine the feature importance of provided data we will be using Predicted target values against number of attributes.



```
tmp = pd.DataFrame({'Feature': predictors, 'Feature importance': clf.feature_impor
tances_})
tmp = tmp.sort_values(by='Feature importance',ascending=False)
plt.figure(figsize = (7,4))
plt.title('Features importance',fontsize=14)
s = sns.barplot(x='Feature',y='Feature importance',data=tmp)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
plt.show()
```

Figure 21. Training AdaBoost

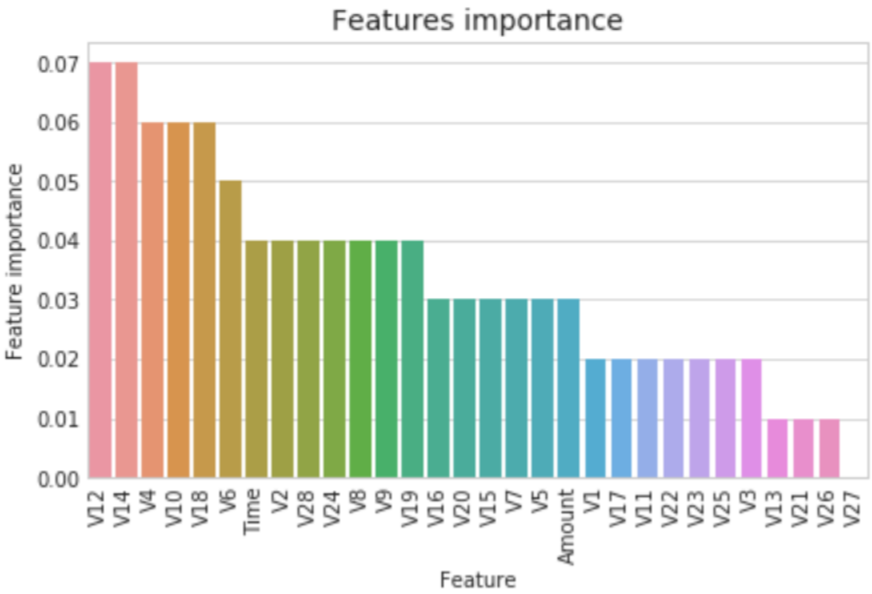


Figure 22. AdaBoost Feature Importance

Above mentioned graph shows that most important and useful features are V12, V14, V4, V10, V18, V6.

6.2.2. Confusion Matrix

We will create another confusion matrix using target and predicted target values for AdaBoost classifier.

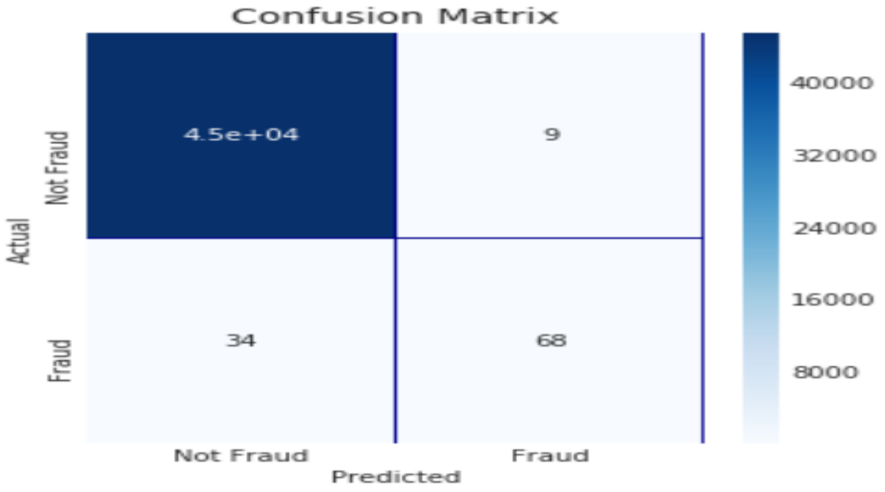


Figure 23. AdaBoost Confusion Matrix

6.2.3. Calculating ROC – AOC Score

To calculate ROC – AOC score we used target values and predicted target values obtained in above steps.

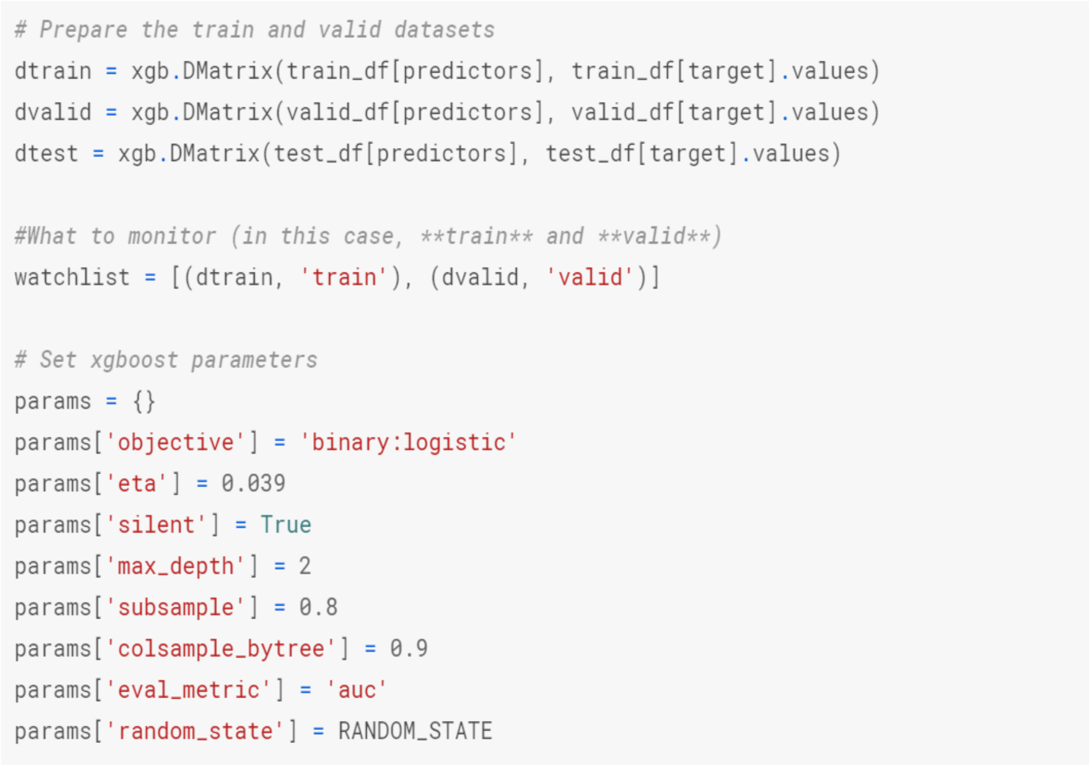
- roc_auc_score(valid_df[target]. values, preds)

The ROC-AUC score obtained with AdaBoost Classifier is 0.83.

6.3. XGBoost Classifier

We will now run XgBoost model using the training set for training. Then, we will use the validation set for validation.

We need to prepare XGBoost Model before training it. We will initialize the DMatrix objects for training and validation, starting from the datasets. We also need to set some of the parameters for the model tuning.



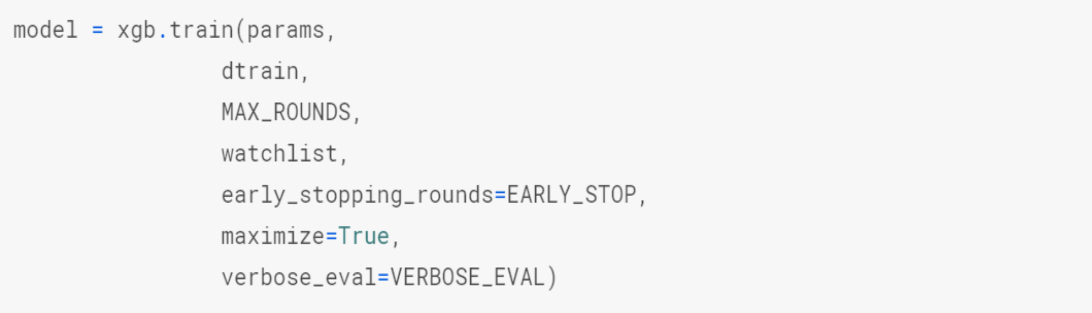
```
# Prepare the train and valid datasets
dtrain = xgb.DMatrix(train_df[predictors], train_df[target].values)
dvalid = xgb.DMatrix(valid_df[predictors], valid_df[target].values)
dtest = xgb.DMatrix(test_df[predictors], test_df[target].values)

#What to monitor (in this case, **train** and **valid**)
watchlist = [(dtrain, 'train'), (dvalid, 'valid')]

# Set xgboost parameters
params = {}
params['objective'] = 'binary:logistic'
params['eta'] = 0.039
params['silent'] = True
params['max_depth'] = 2
params['subsample'] = 0.8
params['colsample_bytree'] = 0.9
params['eval_metric'] = 'auc'
params['random_state'] = RANDOM_STATE
```

Figure 24. XGBoost Preparing Model

After preparing XGBoost Model, we need to train the model using training data.



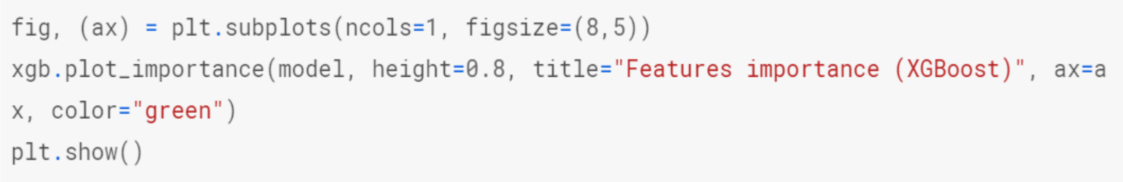
```
model = xgb.train(params,
                dtrain,
                MAX_ROUNDS,
                watchlist,
                early_stopping_rounds=EARLY_STOP,
                maximize=True,
                verbose_eval=VERBOSE_EVAL)
```

Figure 25. XGBoost Training Model

We trained the XGBoost using the dtain and params obtain while creating the model. Next step we followed it to visualize feature importance.

6.3.1. Feature Importance

Feature importance graph can help us better understand the model’s logic, we can verify it being correct but also work on improving the model by focusing only on important features. Determining important features for XGBoost classifiers.



```
fig, (ax) = plt.subplots(ncols=1, figsize=(8,5))
xgb.plot_importance(model, height=0.8, title="Features importance (XGBoost)", ax=a
x, color="green")
plt.show()
```

Figure 26. Determining Feature Importance

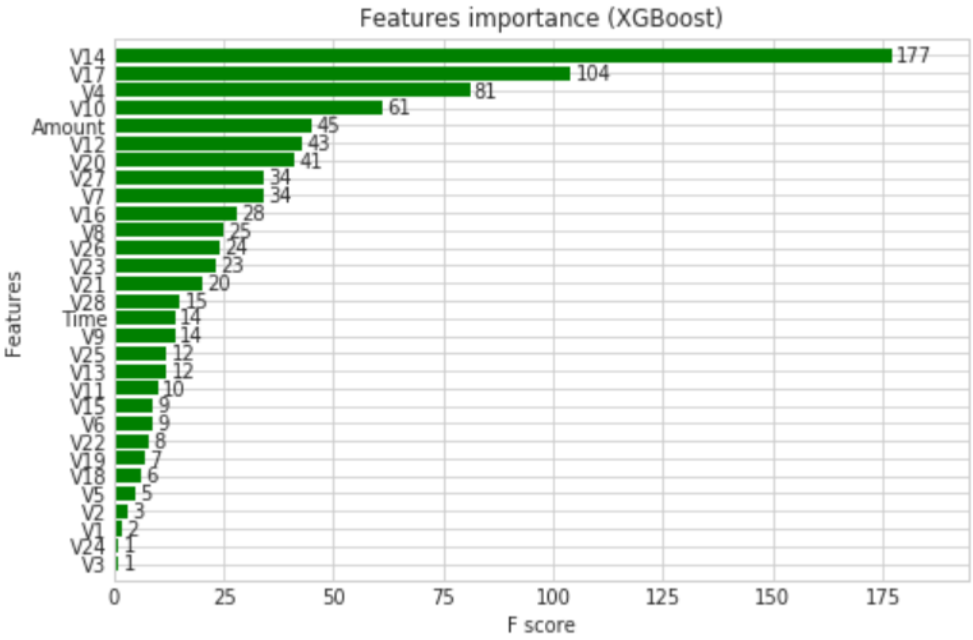


Figure 27. XGBoost Feature Importance

Above mentioned graph shows that most important and useful features are V14, V17, V4, V10, Amount.

6.3.2. Calculating ROC – AOC Score

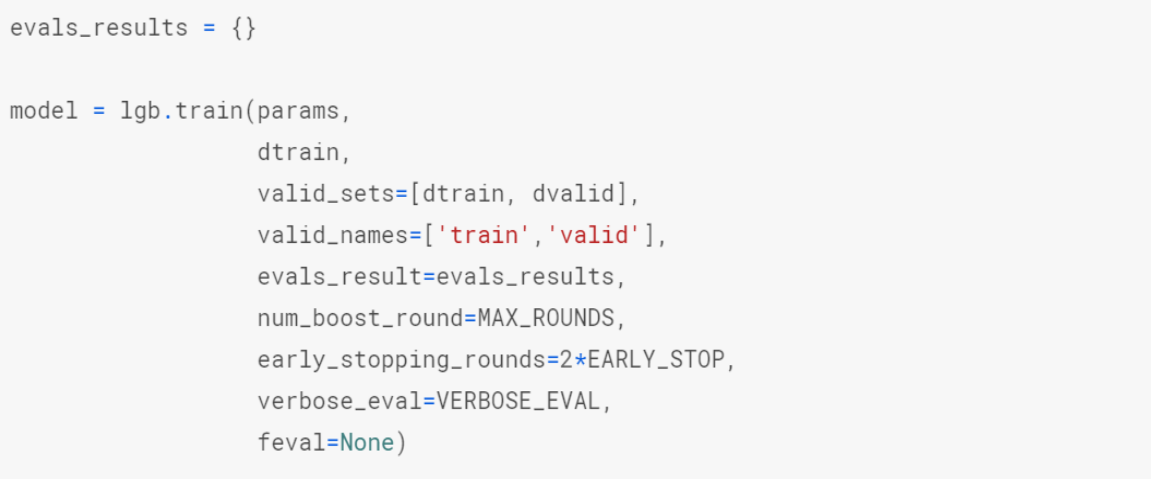
To calculate ROC – AOC score we used target values and predicted target values obtained in above steps.

- roc_auc_score(valid_df[target]. values, preds)

The ROC-AUC score obtained with AdaBoost Classifier is 0.97.

6.4. LightGBM Classifier

Let’s set the parameters for the model. We will use these parameters only for the first lgb model.



```
evals_results = {}

model = lgb.train(params,
                  dtrain,
                  valid_sets=[dtrain, dvalid],
                  valid_names=['train','valid'],
                  evals_result=evals_results,
                  num_boost_round=MAX_ROUNDS,
                  early_stopping_rounds=2*EARLY_STOP,
                  verbose_eval=VERBOSE_EVAL,
                  feval=None)
```

Figure 28. Light GBM Model

We trained the LightGBM using the dtrain data and params which were generated while creating model. Next step we followed it to visualize feature importance.

6.4.1. Calculating ROC – AOC Score

To calculate ROC – AOC score we used target values and predicted target values obtained in above steps.

- roc_auc_score(valid_df[target]. values, preds)

The ROC-AUC score obtained with AdaBoost Classifier is 0.94.

7. CONCLUSION

We investigated the data, checked for data unbalancing, visualized, and understood the relationship between different features. We then used four predictive models to perform validation by splitting dataset into 3 parts, a train set, a validation set and a test set. For the first two models, we only used the train and test set.

- RandomForrestClassifier, provided AUC score of 0.85 by predicting target for the test set.
- We followed with an AdaBoostClassifier model, with lower AUC score (0.83) for prediction of the test set target values.
- We then experimented with a XGBoost model. In this case, se used the validation set for validation of the training model. We used the model with the best training step, to predict target value from the test data; the AUC score obtained was 0.974.
- We then presented the data to a LightGBM model. We used both train-validation split and cross-validation to evaluate the model effectiveness to predict 'Class' value, i.e. detecting if a transaction was fraudulent. For the test set, the score obtained was 0.946.

It shows all simulation attempts are with in AUC [0.0,1.0], since we are successful in interpreting that all classifiers show AUC more than .80. From the experiments the result that has been concluded is that Random Forest Classifier has an accuracy of 85% while AdaBoost shows accuracy of 83%, and LightGBM shows accuracy of 93% but the best results are obtained by XGBoost with a precise accuracy of 97%. The results obtained thus conclude that XGBoost Classifier shows the most precise and high accuracy of 97% in problem of credit card fraud detection with dataset provided by ULB machine learning.

8. REFERENCES

[1] Principal Component Analysis, Wikipedia Page, https://en.wikipedia.org/wiki/Principal_component_analysis

[2] RandomForrestClassifier, http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[3] ROC-AUC characteristic, https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve

[4] AdaBoostClassifier, http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[5] CatBoostClassifier, https://tech.yandex.com/catboost/doc/dg/concepts/pythonreference_catboostclassifier-docpage/

[6] XGBoost Python API Reference, http://xgboost.readthedocs.io/en/latest/python/python_api.html

[7] LightGBM Python implementation, https://github.com/Microsoft/LightGBM/tree/master/python-package

[8] LightGBM algorithm, https://www.microsoft.com/en-us/research/wp-content/uploads/2017/11/lightgbm.pdf

[9] Raj S.B.E., Portia A.A., Analysis on credit card fraud detection methods, Computer, Communication and Electrical Technology International Conference on (ICCCET) (2011), 152-156.

[10] Jain R., Gour B., Dubey S., A hybrid approach for credit card fraud detection using rough set and decision tree technique, International Journal of Computer Applications 139(10) (2016).

[11] Dermala N., Agrawal A.N., Credit card fraud detection using SVM and Reduction of false alarms, International Journal of Innovations in Engineering and Technology (IJIET) 7(2) (2016).

[12] Phua C., Lee V., Smith, Gayler K.R., A comprehensive survey of data mining-based fraud detection research. preprint arXiv:1009.6119 (2010).

[13] Bahnsen A.C., Stojanovic A., Aouada D., Ottersten B., Cost sensitive credit card fraud detection using Bayes minimum risk. 12th International Conference on Machine Learning and Applications (ICMLA) (2013), 333-338.

[14] Carneiro E.M., Dias L.A.V., Da Cunha A.M., Mialaret L.F.S., Cluster analysis and artificial neural networks: A case study in credit card fraud detection, 12th International Conference on Information Technology-New Generations (2015), 122-126.

[15] Hafiz K.T., Aghili S., Zavarsky P., The use of predictive analytics technology to detect credit card fraud in Canada, 11th Iberian Conference on Information Systems and Technologies (CISTI) (2016), 1-6.

[16] Sonepat H.C.E., Bansal M., Survey Paper on Credit Card Fraud Detection, International Journal of Advanced Research in Computer Engineering & Technology 3(3) (2014).

[17] Varre Perantalu K., Bhargav Kiran, Credit card Fraud Detection using Predictive Modeling (2014) Volume 3, Issue 9 IJIRT, ISSN: 2349-6002.

[18] Stolfo S., Fan D.W., Lee W., Prodromidis A., Chan P., Credit card fraud detection using meta-learning: Issues and initial results, AAAI-97 Workshop on Fraud Detection and Risk Management (1997).

[19] Maes S., Tuyls K., Vanschoenwinkel B., Manderick, B., Credit card fraud detection using Bayesian and neural networks Proceedings of the 1st international noise congress on neuro fuzzy technologies (2002), 261-270.

[20] Chan P.K., Stolfo S.J., Toward Scalable Learning with Non- Uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection, In KDD (1998), 164-168.

[21] Rousseeuw P.J., Leroy A.M., Robust regression and outlier detection, John Wiley & sons (2005) ACM Library ISSN: 978-0-471-85233-9.

[22] Wang C.W., Robust automated tumor segmentation on histological and immunohistology chemical tissue images, PloS one 6(2) (2011).

[23] Sait S.Y., Kumar M.S., Murthy H.A. User traffic classification for proxy-server based internet access control, IEEE 6th International Conference on Signal Processing and Communication Systems (ICSPCS) (2012), 1-9.

[24] S. Yang, Stochastic test functions and design optimization using firefly algorithm, International Journal of Bio-Inspired Computation (2010), Vol 2.

[25] Dueck G., Scheur T, A General-Purpose Optimization Algorithm appearing Superior to Simulated Annealing. Journal of Computational Physics (1990), 161–175.

[26] S. Mahfoud and G. Mani, Financial forecasting using genetic algorithms, Applied Artificial Intelligence (1996), Vol 10.

[27] K-S. Shin, Y-J. Lee, A genetic algorithm application in bankruptcy prediction modeling, Expert Systems with Applications (2002), Vol 23.

[28] R. S. Parpinelli, H. S. Lopes and A. A. Frietas, Data Mining with an Ant Colony Optimization Algorithm, IEEE (2002), Vol 6.

[29] M-J. Kim and I. Han, Decision rules from qualitative bankruptcy data using genetic algorithms, Expert Systems with Applications (2003), Vol 25.

[30] T. Sousa, A. Neves, and A. Silva, A particle swarm data miner, 11th Portuguese Conference AI, Workshop on Artificial Life and Evolutionary Algorithms (2003), 43–53.

[31] Y. Liu, Z. Qin, Z. Shi and J. Chen, Rule discovery with particle swarm optimization, Advanced Workshop on Content Computing (2004), Vol 3309.

[32] J. Ji, N. Zhang, C. Liu and N. Zhong, An ant colony optimization algorithm for learning classification rules, in Proc. IEEE/WIC (2006), 1034–1037.

[33] X. Zhao, J. Zeng, Y. Gao and Y. Yang, Particle swarm algorithm for classification rules generation, Proc. of the Intelligent Systems Design and Applications, IEEE (2006), 957–962.

[34] N. Holden and A. A. Frietas, A Hybrid PSO/ACO algorithm for classification, Proc. Genetic and evolution computation conf. (2007), 2745–2750.

[35] H. Su, Y. Yang and L. Zha, Classification rule discovery with DE/QDE algorithm, Expert Systems with Applications (2010), Vol.37.

[36] Aamodt, E. Plaza, Case-based reasoning: foundational issues, methodological variations, and system approaches, Artificial Intelligence Communications 7 (1) (1994) 39–59.