Classifying Bitcoin Ponzi Schemes with Machine Learning

Jarred van de Voort, Sean Coneys

Department on Computer Science, New York University Shanghai, Shanghai, China

December 1, 2018

Abstract
The invention of cryptocurrencies has created new opportunities for criminals to leverage a variety of traditional scam techniques. Fraudulent activity associated with cryptocurrency is experiencing explosive growth and is a significant barrier for widespread adoption of blockchain technology. Early detection of ponzi schemes in this domain is a problem that is a good candidate for machine learning and whose impact surpasses hundreds of millions of dollars each year. This paper extends previous classification models for bitcoin ponzis from start to finish with a focus on dealing with the intrinsic relative imbalance in the data. We explore a variety of balancing techniques as well as models for dealing with the data and discuss their efficacy with regards to this classification problem. Finally we look to the future and discuss interesting avenues of investigation, improvements in data pipelining and data balancing, and the implications of these models now and in the future.

Introduction

Following the first proof of concept of Bitcoin established in 2009 by Satoshi Nakamoto, cryptocurrencies quickly became an attractive option for developers and investors looking to leverage new technology. Cryptocurrencies utilize blockchain ledgers maintained by all members, ensuring that transactions are validated and reliable. The underlying decentralized nature of blockchain ledgers gives users the ability to transfer currency between any two nodes on the network using generated alphanumeric addresses that require no personal information.[1]

Currently, the largest barriers to widespread adoption of the cryptocurrency ecosystem are fraud, transaction processing times, and ease of use. Bitcoin has seen large growth in fraudulent activity due to blockchain crime's low barriers to entry and the difficulty of de-anonymizing transactions[2]. In the first three quarters of 2018 alone, the equivalent of over $927 million USD in cryptocurrency was lost to theft and fraud[3]. Cryptocurrency fraud comes in many different forms each with their own unique patterns and difficulties in identification. Of these, ponzi schemes are an exceptionally well suited candidate for machine learning as they tend to operate on common identifiable patterns[4]. In a ponzi scheme, funds from new investors are used to pay old investors. Once the scheme has reached a point where new funds can no longer be used to pay old investors, the scammer will deposit the remaining funds into a number of addresses. Many of these advertised schemes deceive investors with a facade of fancy marketing and websites, whereas others acknowledge the ponzi scheme with the hope that it will attract short term investors looking to get in and out quickly.

Ponzi schemes are reliant upon new investors to perpetuate the scheme and as such, are forced to advertise. One of the best indicators of a bitcoin ponzis success is its engagement with users[5] . These types of scams are typically advertised as "HYIP" or high yield investment programs that promise overly consistent returns at little to no risk.[6] Most wallet advertising (and aggregation) attempts focus on the two largest bitcoin communities: bitcointalk.org and reddit.com. A recent aggregation of ponzi wallets from the larger of the two (bitcointalk.org) yielded approximately 1800 wallets[5]. Knowing that the majority of bitcoin discussion occurs on these sites and assuming that the largest community yields the most results its not entirely unreasonable to estimate the total number of bitcoin ponzis to be somewhere on the order of 10,000. Estimates for the current number of cryptocurrency wallets suggest in excess of 35 million address with 96% of these supporting bitcoin transactions[7]. This suggests that the data imbalance is both relative and intrinsic. BTC ponzi schemes are not rare in their own right, but are rare relative to the number of total addresses with an estimated class imbalance in excess of 1000:1 (using 10,000 as the estimate for total ponzi schemes and 33,600,000(35,000,000*.96)) as the estimate for total bitcoin wallets the imbalance is approximately 3360:1). Although in some cases of class imbalance the minority class can be learned without balancing[8] the complexity of the data in this problem is such that this is not the case.

A prior study from the University of Cagliari has demonstrated that bitcoin ponzi schemes could be reliably identified using current classification methods[9]. Our data and feature set are largely derived from their research. Their research briefly discusses random undersampling (RUS) and random oversampling(ROS) as well as a deeper discussion on the efficacy of three different models for classifying the data. Our paper intends to extend this research by delving deeper into class balancing discussing methods for random sampling, smart undersampling, synthetic oversampling, cluster based sampling, and combinations of under and oversampling as they pertain to classifying bitcoin ponzi schemes. Additionally, we expand upon their model selection research investigating the efficacy of a variety of different models for this problem.

The Dataset and Features

For feature generation we constructed a three part pipeline in the form of:
Wallet Identification -> Transaction Data Extraction -> Feature Calculation

Wallet Identification

Most sources we found for already identified ponzi wallets had a relatively small number of addresses so we compiled addresses from a number of sources including the Calgary studies dataset, bicoinswhoswho.com (a website where people can report fraudulent wallets), and from manual identification of ponzi schemes from a variety of crypto 'HYIP' forums.

For benign wallets we chose to use the list of approximately 3000 addresses from the Cagliari study[9]. Random sampling of bitcoin wallets does not ensure that a wallet you select is not associated with fraudulent activity so every wallet must be individually vetted. Due to the high time cost of manually compiling an address list we chose to use this set because we believe it to be only non-fraudulent wallets and to compiled recently enough to be representative of all benign BTC wallets in the current ecosystem.

Transaction Data Extraction

The next step in the pipeline is to extract the necessary data to calculate the features for each wallet from the blockchain. This is a fairly complex process and a number of open source tools are available. Ideally we would have used the Cagliari study's[9] data collection tool to replicate their data set, but due to difficulties acquiring the required 200 GB blockchain, we opted to use the blockexplorer.com api. All of our features for a wallet can be calculated given you have the data for each transaction in a wallet, and if the data for each transaction includes a timestamp and a list of all inputs/outputs for the transaction that includes all wallets involved and the associated input/output values. Many blockchain APIs are geared towards traders, inspection of specific transactions, or provide over-simplified or incomplete data for each transaction. Block explorer provided one of the few apis that met all of our requirements for data extraction. Sometimes due to a variety of issues our queries would fail to properly resolve, so we instituted robust error handling that would prevent addresses with errors associated with them from writing to a csv and instead output the address and associated error code to a logfile for either a rerun or a manual inspection. Addresses that generated errors were rerun, but ultimately if an addresses' data could not be fully and correctly parsed, we chose to exclude it from the dataset rather than risk learning on incomplete data.

Feature Extraction

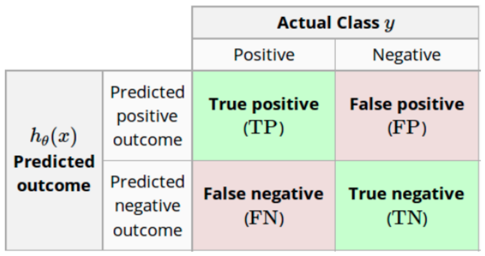
The first step in feature extraction for our data was dividing a wallets transactions into incoming and outgoing transactions. Parsing BTC transactions is rather complicated and an explanation of it is beyond the scope of this paper[10]. To split the transactions into outgoing vs incoming we used a rule based system where if the address of the wallet appeared in the list of inputs it was outgoing and if it had an output to the address of the wallet but no input from the address it was an incoming transaction. After seperation of incoming and outgoing transactions we began feature calculation. Our features are derived from the Cagliari study[9] and are defined as follows:

The lifetime of the address - the number of seconds between the time of the first and last transactions the wallet engaged in
Most active day - The highest number of transactions performed by the wallet in one day
The number of active days - the number of days where at least one transaction in or out occurred
The ratio of transactions in vs transactions out - number of tx_in over the number of tx_out
Number of incoming transactions - the total number of transactions received by an address during its lifetime
Number of outgoing transactions - the total number of transactions sent by an address during its lifetime
Number of addresses that received from and sent to from address- number of unique addresses that both sent money to and received money from the account
Median delay - the median delay in seconds between the time the wallet received its last incoming transaction and then performed an outgoing one
Mean delay - the mean delay in seconds between the time the wallet received its last incoming transaction and then performed an outgoing one
Minimum delay - the shortest delay in seconds between the time the wallet received its last incoming transaction and then performed an outgoing one
Maximum delay - the longest delay in seconds between the time the wallet received its last incoming transaction and then performed an outgoing one
Total received - The total amount of BTC received by the address
Total sent - The total amount of BTC sent by the address
Average value in - Average size of the output to the address in an incoming transaction
Average value out - Average size of the address' total inputs into an outgoing transaction minus the total of the outputs back to the address
Maximum Difference - The maximum difference between the balance of the address in a two day window. We used a discrete window from the start of day one to to the end of day two as opposed to a moving 48 hour time interval

Here again we institute error handling where all addresses where we are unable to fully calculate the features are prevented from being written and are instead written with an error code to a log for inspection.

Methods

Metrics
Discussing the results of classification models on imbalanced data requires extra consideration when specifying metrics used to determine the success of a model. A naive approach using accuracy as a metric would find consistently high scores,due to the model predicting the majority class for all data points. Instead, we'll use recall as our metric for determining the performance of our model. Since there is a high cost associated with false negatives in this case(a false negative here means a failure to classify a ponzi scheme as a ponzi) recall is a much more indicative metric of a model's performance. To account for the tradeoffs between precision and recall in our models we also asses models based on the area under curve(AUC) for a precision-recall curve[8]. The confusion matrix below illustrates how these metrics are associated with classifying ponzi schemes[11]

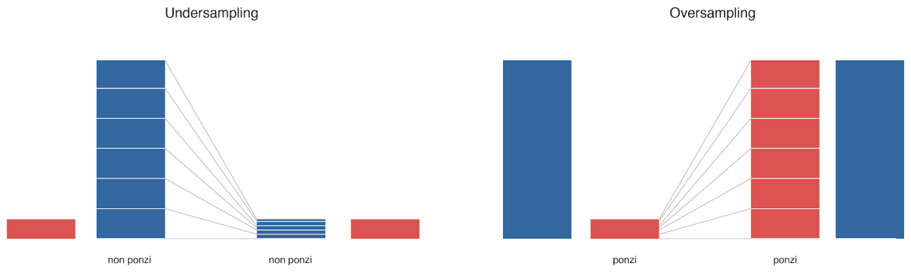


		Actual Class y	
		Positive	Negative
$h_\theta(x)$ **Predicted outcome**	Predicted positive outcome	**True positive** (TP)	**False positive** (FP)
	Predicted negative outcome	**False negative** (FN)	**True negative** (TN)

$TP = Correctly\ identified\ ponzi$
$FP = Incorrectly\ identified\ non - ponzi\ as\ ponzi$
$FN = Incorrectly\ identified\ ponzi\ as\ non - ponzi$
$TN = Correctly\ identified\ non - ponzi$

$$Accuracy = \frac{tp+tn}{tp+tn+fp+fn} \quad Precision = \frac{tp}{tp+fp} \quad Recall = \frac{tp}{tp+fn}$$

Class Balancing
Basic resampling techniques fall under two main categories, undersampling and oversampling. In undersampling, data points are extracted from the majority class to create a new dataset for the majority class similar in size to the minority class. For oversampling the inverse is performed. Data points from the minority class are duplicated to create a new dataset where the minority class is of similar size to the majority class.



Due to the reduction of data provided to the model for learning undersampling is prone to underfitting. Similarly, due to the duplication of data points oversampling is prone to overfitting the data[8].

Near Miss
To combat the reduction in data provided to the model in undersampling the near-miss algorithm was created. In near miss sampling the data points from the majority set are chosen based off of their distance from the other data points as computed by k-nearest neighbors. By choosing points with small distances to their nearest neighbors the chosen points are theoretically more indicative of the dataset as a whole[8].

SMOTE - Synthetic Minority Over-sampling Technique
In order to address the overfitting problem created by datapoint duplication in random over sampling SMOTE synthesizes new data points for the minority class by extrapolating features based on prior minority class instances and its k nearest neighbors. As a result, the new generated instances cause the classifier to expand the decision region for the minority class. The algorithm first obtains the k-nearest neighbors of a sample of the minority class x_o, then randomly select a minority class sample amongst k nearest neighbors $\bar{x}_i$. Using these two samples, the algorithm then generates a new instance by interpolating between the two[12]:

$$x_{new} = x_o + rand(0,1) \times (x_i - x_o)$$

Centroid Clustering
Centroid Cluster balancing performs undersampling by generating new data points using the k-means algorithm where k is equal to the number of datapoints in the minority class. A new datapoint is generated for each cluster by averaging the features of all majority class data points in the cluster[8].

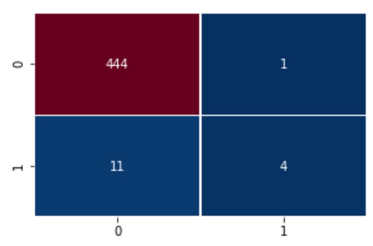
Smote Tomek
The SMOTE tomek algorithm first performs traditional SMOTE. Given a pair of samples $(x_i, x_j)$from the dataset: Given, $x_i \in S_{min}, x_j \in S_{maj}$, the pair is considered a Tomek link if there is no x_k such that $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j x_k) < d(x_i, x_j)$.If a set of two samples satisfy one of these conditions, one of the samples is noise or is near a border [8][13]. Tokmek links identify cases where either one of the data points is noise or they both are borderline [8][13]. After the tomek links are identified they can be used to perform either undersampling or data cleaning[13]. In the case of undersampling only instances of the majority class identified by the tomek links are removed, in the data cleaning approach instances of both classes identified by the links are removed. The implementation of SMOTE tomek used in this study follows the data cleaning approach.

Model Selection
We chose a variety of types of classifiers to give us a more comprehensive overview of performance given the underlying mechanisms that each algorithm operates on. Our model selection is as follows:

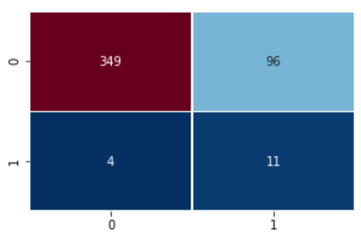
Logistic Regression (LR) - used as a baseline for understanding model performance
Support Vector Machine (SVM) - Allows us examine the effect of different kernels on our dataset
ADA Boosted Classifier (ADA) - tends to be sensitive to noisy data and outliers, but less prone to overfitting
Quadratic Discriminant Analysis Classifier (QDA) - bayesian method that operates on assumption that features are normally distributed
Random Forest Classifier (RF) - has shown high levels of success in recent years with an algorithm that derives averages from series of decision trees to produce best model
Gradient Boosting Classifier (GB) - Gradient boosting algorithms proven to be strong candidates in the machine learning community by utilizing ensemble methods

Results
We first used basic logistic regression to benchmark the performance of oversampling and undersampling during training. As reflected in the confusion matrices, resampling our dataset provided considerable advantages.

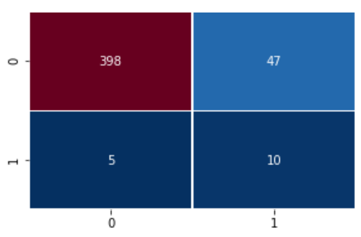
No Sampling



Random Undersampling



SMOTE Oversampling



Diving deeper, we see that without resampling our dataset, our model was only able to identify four out of fifteen ponzi schemes, indicating that our model is potentially underfitting due to class imbalance. By transforming our dataset using undersampling and oversampling techniques, we drastically improved recall and were able to identify most ponzi schemes at the cost of a higher amount of false positives. It's important to note that the cost of a false negative is much higher than that of a false positive, so we can excuse a higher level of false positive. As we'll see when we compare models, this isn't always the case, as some models will elicit a high level of recall simply by classifying all instances as ponzi.

Having demonstrated the effect of resampling, we tested several popular classification models using different resampling techniques. We then generated a performance matrix for our metrics of recall and PRC AUC:

Recall Performance Matrix

	None	RUS	NM	CC	ROS	SMOTE	SNN	STK
LR	0.27	0.73	0.67	0.6	0.73	0.67	0.73	0.67
SVM	0.067	0.73	0.8	0.73	0.6	0.6	0.67	0.6
ADA	0.33	0.8	1	0.93	0.73	0.67	0.73	0.67
QDA	0.73	0.87	1	1	1	0.93	0.93	0.93
RF	0.4	0.73	1	0.93	0.33	0.53	0.73	0.6
GB	0.4	0.8	0.93	0.93	0.6	0.73	0.73	0.67

PRC AUC Performance Matrix

	None	RUS	NM	CC	ROS	SMOTE	SNN	STK
LR	0.59	0.48	0.24	0.53	0.58	0.59	0.61	0.59
SVM	0.54	0.45	0.17	0.55	0.48	0.52	0.51	0.52
ADA	0.52	0.41	0.25	0.53	0.57	0.55	0.61	0.47
QDA	0.51	0.4	0.36	0.52	0.53	0.56	0.55	0.55
RF	0.68	0.51	0.061	0.35	0.57	0.45	0.49	0.44
GB	0.6	0.5	0.25	0.51	0.59	0.53	0.49	0.6

From our recall matrix, we see that the QDA, ADA, and RF models were able to correctly identify every ponzi scheme instance. However, the PRC AUC for instances with perfect recall tells us that this is often at the expense of precision. For example, the RF model using Near Miss resampling was able to detect all ponzi schemes but with a PRC AUC of .061, which is by far the worst level of PRC AUC performance. This is because the RF/NM model classified nearly every instance as a ponzi scheme, rendering a high recall with almost no precision. Therefore, it's important that we balance a high level of recall with a moderate to high level PRC AUC.

Looking at the performance of resampling techniques, we see that Centroid Clustering produces consistently high levels of recall without sacrificing precision. On the other hand, Near Miss undersampling is seen to generate the highest levels of recall at the cost of the lowest overall PRC AUC. Since Near Miss is most effective when samples are more tightly clustered, it's likely that the high level of variance in our non ponzi features diminishes the overall efficiency of this resampling technique.

We chose QDA and RF as strong candidates for further exploration, as they yield the most consistent, high performing models. High levels of recall might indicate that these models might be prone to overfitting, so we'll use cross validation to examine how well our model generalizes. Generally speaking, overfitting tends to occur when there is a large difference between training scores and the cross validation scores due to high variance, whereas underfitting is present when both our training and cross-validation scores are both low. We tuned both models using the most effective hyperparameters for high levels of recall, and compared it to a cross validated recall score with respect to each resampling technique.

QDA Train vs. CV Recall

	RUS	NM	CC	ROS	SMOTE	SNN	STK
Train Rec	0.87	1.00	1.00	1.00	0.87	0.87	0.87
CV Rec	0.88	0.87	0.94	0.76	0.65	0.72	0.65

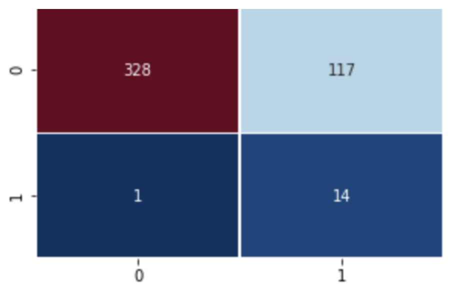
RF Train vs. CV Recall

	RUS	NM	CC	ROS	SMOTE	SNN	STK
Train Rec	0.73	0.80	0.93	0.73	0.73	0.80	0.73
CV Rec	0.85	0.84	0.88	0.67	0.73	0.76	0.69

As suspected, the large difference in recall between our QDA trained set and our cross validated set indicates that the model is prone to overfitting. On the other hand, RF shows a pretty consistent recall metric across both sets demonstrating that it will generalize quite well.

Conclusion
Through our process of cross validation, we were able to derive the best model/resampling combination that consistently produces a yields recall without overfitting. This model is the Random Forest Classifier utilizing the Centroid Clustering resampling technique.

Random Forest with Centroid Clustering



```
Recall:    0.93
Precision: 0.10
AUC:       0.44
```

The model is able to correctly identify fourteen out of fifteen ponzi schemes, while incorrectly classifying a moderate amount of non-ponzi addresses. In the context of bitcoin, this model would provide the FBI, or any kind of regulatory agency a list of addresses for further inspection that would consistently be able to identity actual ponzi schemes.

Future Work and Implications
One of the main difficulties of this project has been data acquisition. Namely, in data labeling and processing. Labeling currently is done mainly through rule based web scraping of forum posts and manual inspection[5][9]. We believe that a multinomial naive bayes model could be a good candidate to better identify forum post advertising ponzis and the related addresses. Data processing in this area is also a large obstacle however, unlike most projects all the data is publicly available in its entirety. The data is very esoteric and requires a deep understanding of how BTC functions in order to parse it into meaningful features. We believe that streamlining the data aggregation process is a very important step in increasing the accuracy of existing models. Strides have been made in tool design for blockchain data aggregation, however these tools require users to have the entire blockchain. As advancements are made in transaction processing on the blockchain, blockchain sizes will greatly increase meriting further study into efficient aggregation tools.

Currently, many models for classifying ponzis on the blockchain are based off of mature or completed ponzi schemes. The classifications provided by these models are only as useful as the ability to prevent ponzi fraud that they provide, therefore the ultimate goal should be the detection of immature or developing ponzi schemes. We believe that a neural network that accepts transaction data associated with wallets instead of pre-computed features could be a good candidate for this type of problem. The classification models trained on mature ponzi's could be used in addition to the posting classification model previously discussed to help identify and label likely ponzi schemes that could then be fed into the neural network.

An interesting point to note is that even if ponzi schemes can be accurately identified by a model the impact may be negligible for personal use. Many ponzis generate wallets on demand for users to deposit into. The money is then transferred from the generated wallet to a central wallet. A user could only view the wallet they are supposed to deposit into. So if the address was ran in the model it would have no transactions and would certainly be classified as a non-ponzi, giving the user no indication as to whether or not they are investing into a ponzi scheme. The model could of course still detect the central wallet if all transaction data on the blockchain is fed into it, which could help law enforcement agencies identify wallets that require action.

An important aspect of bitcoin ponzi fraud that we were unable to adequately investigate is the clustering of wallets in ponzi schemes. If the vast majority of transactions into a wallet come from newly created wallets it could be indicative that they are generating wallets on demand for deposits. Additionally, sometimes money is moved around between a number of wallets for security, technical, or obfuscation purposes. Investigating how money moves once it has been sent into a ponzi scheme has great potential for exposing more about the inner workings of these schemes for identifying potential new features and for identifying wallets that are withdrawn from by, and can be linked to the perpetrators.

This paper was unable to cover all class balancing data techniques, we believe this paper to be a good baseline analysis of sampling based techniques but further research on non-sampling based techniques and kernel based techniques for svms is merited.

References

[1] Nakamoto, S., Bitcoin: A Peer-to-Peer Electronic Cash System. Available at: https://bitcoin.org/bitcoin.pdf.
[2]Wieczner, J. (2017) 'The 21St-Century Bank Robbery', Fortune, 176(3), pp. 52–59. Available at: http://proxy.library.nyu.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=bah&AN=124711446&site=ehost-live (Accessed: 6 December 2018).
[3]CipherTrace, 2018. Cryptocurrency Anti-Money Laundering Report.
[4]Artzrouni, M., The mathematics of Ponzi schemes. Mathematical Social Sciences, (58), pp.190–201. Available at: https://ac.els-cdn.com/S0165489609000572/1-s2.0-S0165489609000572-main.pdf?_tid=dcaec7d7-0098-4146-9bcf-8ea212f45a75&acdnat=1544110392_3834c00f898bb54be022ba77322628f3 [Accessed December 6, 2018].
[5]Vasek , M. & Moore, T., 2018. Analyzing the Bitcoin Ponzi Scheme Ecosystem. Bitcoin Workshop. Available at: https://fc18.ifca.ai/bitcoin/papers/bitcoin18-final11.pdf.
[6]Moore, T., Han, J. & Clayton, R., 2012. The Postmodern Ponzi Scheme: Empirical Analysis of High-Yield Investment Programs. Financial Cryptography and Data Security Lecture Notes in Computer Science, pp.41–56.
[7]Hileman, G. & Rauchs, M., 2017. Global Cryptocurrency Benchmarking Study
[8]He, H. & Garcia, E.A., 2009. Learning from Imbalanced Data. IEEE Transaction On Knowledge and Data Engineering, 21(9). Available at: http://www.ele.uri.edu/faculty/he/PDFfiles/ImbalancedLearning.pdf.
[9]Bartoletti, M., Pes, B. & Serusi, S., 2018. Data Mining for Detecting Bitcoin Ponzi Schemes. 2018 Crypto Valley Conference on Blockchain Technology (CVCBT).
[10]Anon, Transaction. Bitcoin Wiki. https://en.bitcoin.it/wiki/Transaction
[11]Grigorev, A., 2018. Supervised Learning: Classification and Regression. Packt Hub. Available at: https://hub.packtpub.com/supervised-learning-classification-and-regression/ [Accessed December 6, 2018].
[12]Hu, F. & Li, H., 2013. A Novel Boundary Oversampling Algorithm Based on Neighborhood Rough Set Model: NRSBoundary-SMOTE. Mathematical Problems in Engineering, 2013, pp.1–10.
[13]Gustavo E. A. P. A. Batista, Prati, R.C. & Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1), p.20.