ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

Credit Fraud Prediction Using Machine Learning Algorithms

Yuan Huang[1], Ya-Chi Lee[2], Zongmian Huang[3], Zongxi Huang[4], Zixi Zhong[5], Zihao Jiang[6]

[1]Chongqing University of Technology, Chongqing 400054, China

[2]University of California Irvine, CA, 92617, US

[3]Pepperdine University, Malibu, CA, 90263, US

[4]Pepperdine University, Malibu, CA, 90263, US

[5]South China University of Technology, Guangzhou 510641, China

[6]Nanjing Foreign Language School Xianlin Campus, Nanjing, Jiangsu 210046, China

Abstract

While the use of credit card payment is becoming more and more popular, the rise of credit card frauds also becomes a highly notable issue and causes serious problems in many financial industries. This paper presents a complete process of a fraud model building project that includes the discussion of data preparation, data cleaning, feature selection, model building, and corresponding results. The fraud models built in this project are supervised models with given labels on data to indicate whether it is fraud or non-fraud. In more specifically, for feature selection, starting with 308 expert variables, the univariate filter methods KS and FDR are applied to cut off the majority of variables and to keep 80 of them, and the wrapper is applied to reduce the number of variables to 30 using backward selection with the logistic regression model. Also, in the model algorithms section, the performance and result of logistic regression, neural network, boosted tree, and random forest is compared and illustrated. In this paper, neural net is determined to be the best predictive model with out-of-time FDR of 53.7091%.

Keywords

Credit Fraud Prediction; Machine learning; Data preparation; Data cleaning; Feature selection; Model building.

1. INTRODUCTION

According to the research statistics, the percentage of Americans with a credit card rose once again in the first quarter of 2019 to 60.5% [1]. As credit card becomes dominant payment methods around the world, the credit card frauds also become a global issue and frequently happen in daily life. Card frauds take place in various ways, one of the most representative examples is to conduct the credit card fraud by the theft of an actual card. By illegally obtaining and using the cardholder's personal information and bank account, the embezzled card could be used to conduct purchasing or other fraudulent behaviors to obtain different degrees of financial gain. The prevailing use of payment cards along with the frequent occurrence of these fraudulent phenomena reflects the trade-off between convenience and fraud.

Generally, a fraud transaction is characterized by anomaly information or data showing up on transactional records. To combat the fraud issue given the historical transaction data with fraud labels on it, supervised models are needed to analyze and train on the collected historical data

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

and therefore be capable of identifying the fraudulent transactions from the new transactional data [2-7]. The project presented in this paper shows different supervised model algorithms including logistic regression, neural network, boosted tree, and random forest for model building, and their performances are compared to find the one that has the best fraud detection rate [8-13]. The model with outstanding performance could then be used as the fraud detection model.

This paper discusses credit card identity fraud prediction using machine learning algorithms. Section 2 Data Description describes the dataset available for this research project. Section 3 Data Cleaning illustrates the process of handling exclusion, outliers, missing fields, and frivolous field values. Section 4 Variable Creation describes the formulas and mathematical logic for creating all the candidate inputs to model algorithms. Section 5 Feature Selection describes the methods applied to select the variables with useful information from expert variables and presents the list of final variables. Section 6 Model Algorithms includes explanations for all algorithms tired and presents high-level results for each algorithm (namely, Fraud Detection Rate for training, testing, out of time data). Section 7 Model Result presents the result of final optimal algorithm and parameter selection, for training, testing, and out of time populations, and the fraud savings plot. Section 8 summarizes the research and provides recommendations for further research.

2. DATA DESCRIPTION

The data contains 1,000,000 credit card and cell phone applications. To avoid privacy issues, it is generated from real-life U.S. applications over 10 years by synthetic data algorithm to have similar statistical properties. Since our data and the real data share the frequency of occurrence of fields and field relationships, we are able to investigate and build models on these synthetic data. Table 1 shows the summary of the data fields. Each application has ten features, which include:

record: a unique identifier for each application

date: the specific application date

SSN: a nine digits unique U.S. citizen identifier

firstname: the first name of the application user.

last name: the last name of the application user.

address: the address of the application user.

zip5: a five digits US Postal Service ZIP code of the user's address.

dob: the date of birth of the user.

homephone: a ten digits user's home phone number

fraud label: the label that indicates whether the application is a fraud, where "1" represents fraud.

There are three things that need to be noticed. First, our data is sorted by the date of the transaction with the earliest date at the front, thus the record can also be viewed as the time order. Second, the data is synthesized from the real one, thus fields, which includes SSN, zip5, and homephone, may not strictly follow their real-life format. It is caused by leading zero and we will handle it in data cleaning part. Third, we investigate identity fraud in this paper, thus our features are all sensitive and personal identity fields.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

Table 1. Data frame Head

Fields Name	Type	# Unique Values	# Nonfrivolous Records	Nonfrivolous Previous Populated	Mode
record	Index	1,000,000	1,000,000	100%	1
date	Time	365	1,000,000	100%	20160101
ssn	Number	835,819	983,065	98.3%	123456789
firstname	String	78,136	1,000,000	100%	FIRSTN
lastname	String	177,001	1,000,000	100%	LASTN
address	String	828,774	998,921	99.9%	123 MAIN ST
Zip5	Category	26,370	1,000,000	100%	92415
dob	String	42,673	873,432	87.3%	19250101
homephone	Number	28,244	921,488	92.1%	1234567890
Fraud_label	Category	2	1,000,000	100%	0

2.1. Data Field: "Date"

In the process of organizing the data, we found an anomaly data in the date field, which is exhibited on Figure 1. Figure 1 categorizes different transactions into different days according to their date field. There is an apparent outlier at the end of February. The outlier is caused by the fact that not every February has 29 days.

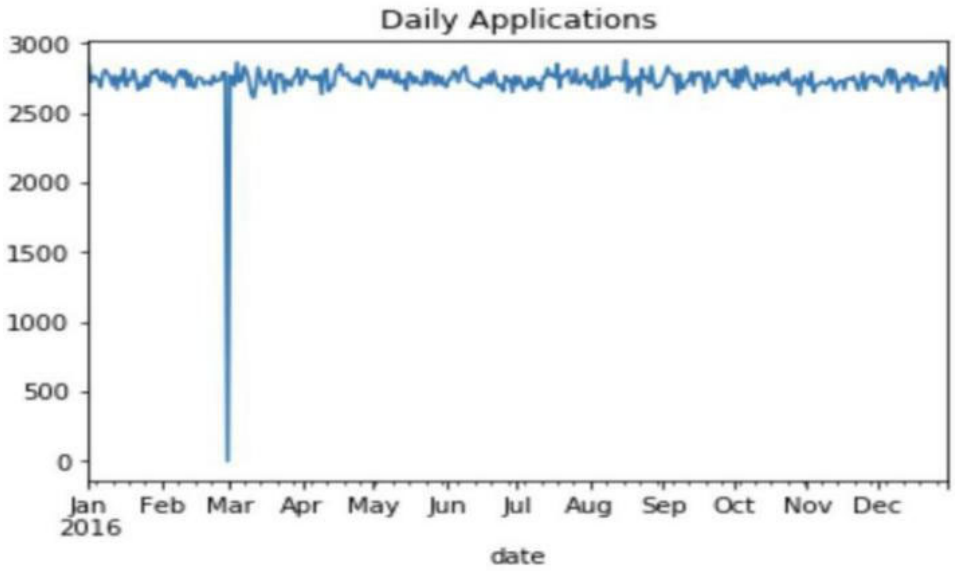

Figure 1. Number of Applications per date

2.2. Data Field: "ssn"

The field of SSN also displays an unusual distribution. There are 16,935 or 1.7% of applications have the number "999999999" in their SSN field. As shown in Figure 2, the number "999999999" occupies a large proportion and are much higher than another SSN number. Table 2 further lists the abnormal SSN in detail. As Table 2 indicates, none of these data from the top 10 is classified as fraud. In addition, only 0.56% of these data are labeled as fraud. They are frivolous elements that can be caused by default input in respond to the customer input nothing. To avoid their influence on the outcome of our algorithms, we will handle these frivolous elements.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

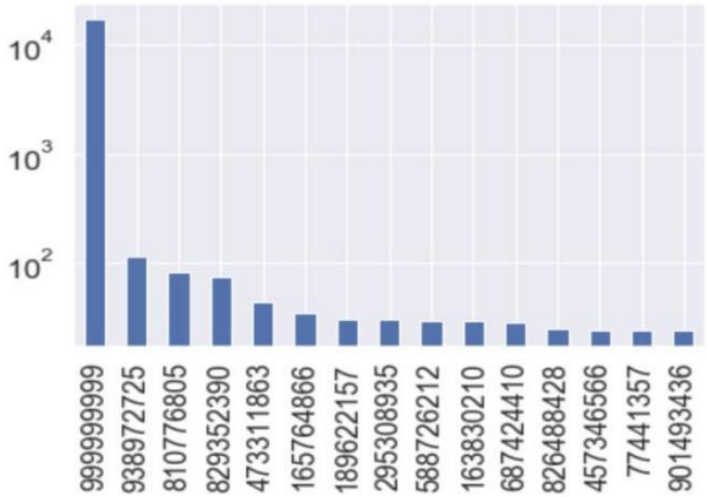

Figure 2. Distribution of SSN

Table 2. Details of some abnormal SSN

record	date	ssn	firstname	lastname	address	Zip5	dob	homephone	fraud_label
11	20160101	999999999	UZZSMXSEE	USJZUSA	726 UMTXU ST	92129	19400126	3026547212	0
12	20160101	999999999	ETXMUSZEM	UJMAMEU	3387 STRUM WY	45982	19580830	5568704443	0
64	20160101	999999999	STUMMAMTS	SSJXTUJM	6393 UEJEA LN	41640	19070626	9999999999	0
68	20160101	999999999	UUZAJXZMT	SRAAMZXU	2557 URRXS DR	98407	20110831	5490098836	0
74	20160101	999999999	AUATMMZX	UUUTAZR	30 ETXRM ST	38402	19760625	2382673773	0
283	20160101	999999999	MSJUZSMZJ	UXMTEZUT	356 UZSRE AVE	74187	19070626	153492955	0
380	20160101	999999999	XRJSMRRRT	RMRXAMX	9769 UAMZJ DR	52351	19070626	3181686949	0
420	20160101	999999999	XRXMTMZEZ	SZRXXJMX	5854 RXRXX PL	34957	19970721	316681200	0
446	20160101	999999999	RRZUZUSX	ESZRRRAA	8970 AXME RD	48333	19110729	5350782211	0
468	20160101	999999999	XETRJXESR	SSJXTUJM	6995 RMSST CT	48162	19700418	2503172355	0

2.3. Data Field: "Address", "Dob"

The address field, homephone field, and dob field have similar issues of frivolous elements. Figure 3 and Figure 4 show the corresponding distribution of the address and home phone number. In the field of homephone, 78512 data have "999999999" as their input; in the field of address, 1079 data are "123 MAIN ST." We also handle these fields later.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

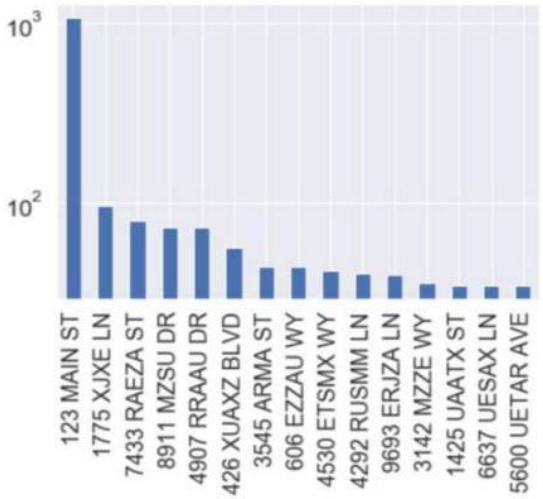

Figure 3. Distribution of address

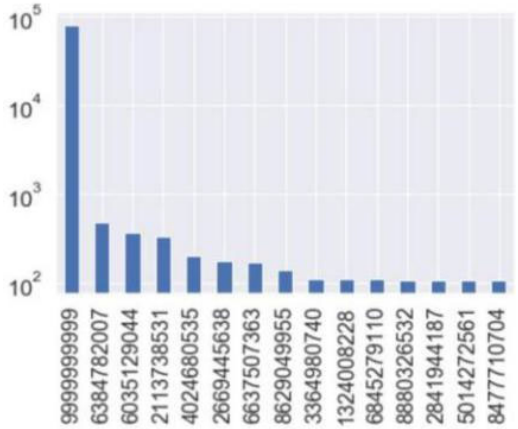

Figure 4. Distribution of phone number

3. DATA CLEANING

Sometimes when the application has missing fields, the business that receive the application usually have some processes that would allow the application actually go through the system. For the data used in this project, some dummy values (frivolous ID elements) for either phone number or social security number are used to fill in the missing fields. Therefore, there are no missing fields presented in the data used in this model building project. However, using dummy values causes problems in variable creation since the frivolous identity element would have a very high counts when there is a variable counting how frequent of that frivolous ID element being seen on a application or data set. This problem might bring confusion to the modeling algorithms because that identity element value is not truly happened that much of time; it is just the dummy values are being used frequently. So, to avoid having unusual values in created variables, the data need to be cleaned through statistical analysis so that they can be neutralized and be normally used in modeling algorithms. Besides, there are outliers need to be identified and further adjusted in the data so that the variables can be created and used correctly in model building.

4. VARIABLE CREATION

In machine learning, model's goodness depends on the data. One should always focus on creating a dataset that is optimized to maximize the information density of the data. Variable

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

creation, also known as feature engineering or variable encoding, is the process of constructing thoughtful inputs to a model and is the most crucial part of building a machine learning algorithm that solves practical problems. Typically, one examines the fields in the raw data, does analysis, cleaning, standardization, and then transformations and combinations of these fields to create special variables that are candidate inputs to models. Raw fields from the raw dataset needs to be first converted into normalized scaled numeric variables. Then, one can build special, explicit variables that best relate the raw fields to the output, using different combinations of fields. This paper uses 10 fields to build 634 candidate variables for models in various combination of them. We first create two new fields by concatenating these fundamental fields together. The two concatenations are presented below:

1. namedob = firstname_lastname_dob
2. fulladdress = address_zip

The namedob is a clever combination since it is close to a unique identifier for a person. It is a rare incident that two individuals with same first and last name have a same date of birth. Fulladdress is another good concatenation of fields as attaching a zip code to an address affords a higher chance to make it a unique address identifier. Then, we identify the fundamental entities that can be used for linking fields together:

ssn, fulladdress, namedob, phone.

These key entities can further build combination groups:

(ssn, fulladdress), (ssn, namedob), (ssn, phone),

(fulladdress, namedob), (fulladdress, phone), (namedob, phone),

(firstname, ssn), (lastname, ssn) ...

New variables are built around these combination groups and linking entities. To create variables that best associate with fraudulent behaviors, we interviewed domain experts {name of advisor?}and fully understood the dynamics of identity fraud and its associating variables. Some indications of identity fraud can be unusual number of applications with the same SSN, address, phone number, name_DOB, etc or a particular (SSN, Name_DOB) being used with many different (addresses, phone). We build expert variables that quantify these conditions.

This paper includes four general types of expert variables:

1. Velocity variables. These variables measure the number of records seen over the past n days, n = {0, 1, 3, 7, 14, 30} days. In another word, how many times an entity or combination group seen over the past n days. A considerably large number of records existing in the recent past transaction can be a strong indication of fraudulent behaviors.

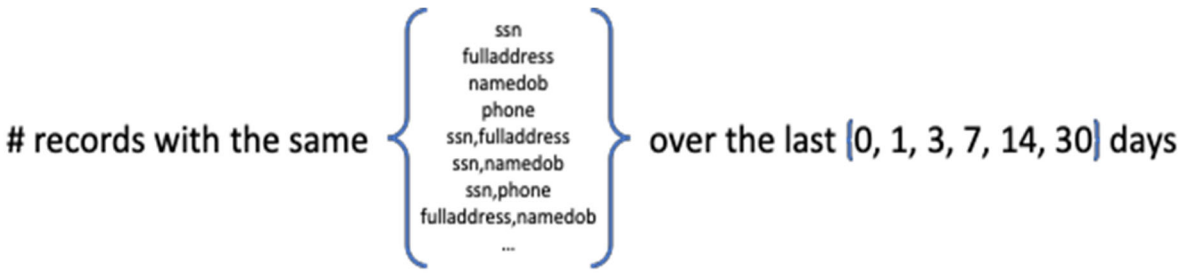

Figure 5. Velocity variables

2. Day since variables. This is another frequency variable that measure how many days since we last saw an entity or combination group.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

3. Relative velocity variables. These variables measure the number of times an entity or record seen in the recent past (past 2 days) compared to the number of times the same entity or record seen over a different time window (past 14 days).

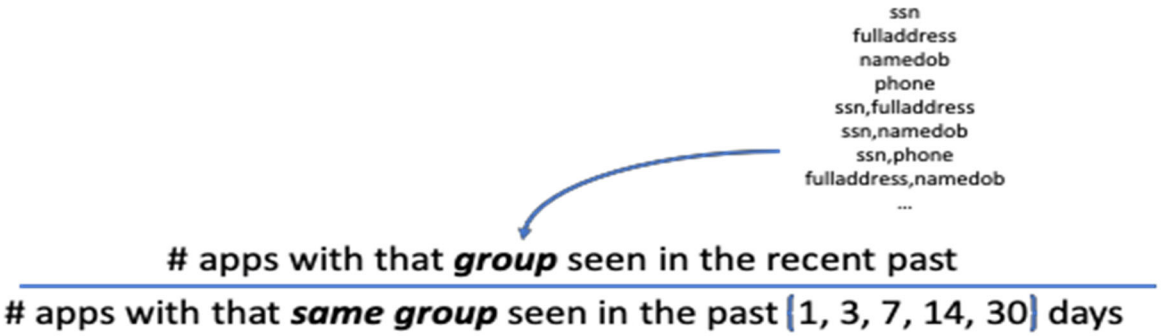

Figure 6. Relative velocity variables

Through constructing meaningful candidate inputs to the machine learning models, we eventually created 248 velocity variables, 14 day since variables, and 372 relative velocity variables.

5. FEATURE SELECTION

The abundance of high dimensional data in various applications poses challenges to machine learning model building process. In a high dimensional dataset, data becomes sparse quickly, meaning there will be separation between data points or empty regions with missing data value. As dimensionality increases, all data points tend to become outliers. Moreover, we will need exponentially more data to see true nonlinearities of the distribution rather than noise in high dimensional data. With the considerable difficulty arises from the curse of high dimensionality, feature selection becomes a crucial step in a model building process. It not only helps reduce dimensionality, but also help select the particular input variables with substantial information and exclude the other useless variables. Good feature selection can help prevent low predictive capability or redundant features from the original group of the features, increase model efficiency, and is the prerequisite for subsequent works [14].

In general, there are three general ways of categorizing feature selection methods (listed in ascending order of how CPU-expensive they are): Filter, Wrapper, and Embedded. A filter is a selecting methodology that is independent of any modeling method. It applies mathematical measure to each column of data independently without building model. Some commonly used mathematical measures for binary classification (fraud = 1, non-fraud = 0) include Pearson correlations, Fisher Score, univariate Kolmogorov-Smirnov, and univariate fraud detection rate. The wrapper method uses a model "wrapped" around the feature selection, and the standard wrapper methods are forward selection, backward selection, or stepwise selection. The embedded method does feature selection as the model is built, and it runs inside the modeling algorithm. Decision trees and the use of regularization are both classic examples of using embedded feature selection.

5.1. Filter Method

The methods applied in this paper are filter and wrapper. The particular mathematical measures applied in filter are univariate Kolmogorov-Smirnov (KS) and univariate fraud detection rate (FDR) at 3%. Univariate KS is a neat measure of distance between two distributions. In the two-class classification problem of fraud prediction, we make separate normalized distributions for the two populations and use KS as the measure for how separate

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

are these two curves. The amount of separation between the distributions indicates the importance of the variable. The more different the curves, the better the variable for separating, and hence the more important the variable. We typically calculate the univariate KS for each variable and abandon the variables with low KS. The Kolmogorov-Smirnov is presented below:

$$KS = \max_{x} \int_{x_{\min}}^{x} \left[P_{good} - P_{bad} \right] dx \tag{1}$$

$$KS = \max_{x} \sum_{x_{\min}}^{x} \left[P_{good} - P_{bad} \right] \tag{2}$$

The first KS formula is represented in integral format and designated for continuous variables. The second KS formula that is represented in sum format can be used for discrete variables. Fraud detection rate measures the percentage of total frauds caught at a particular examination cutoff location. For example, FDR 30% at 5% means the model catches 30% of all the frauds in 3% of the population. This measure indicates how well the candidate variables separate the frauds from the non-frauds. The variables are sorted separately by KS and FDR and then combined together with equal weight using average rank-ordering list to sort variables. This final rank-ordering list of variables represent the results from filter.

5.2. Wrapper Method

The main distinction of a Wrapper method is it has a model "wrapped" around the process, and any modeling algorithm can be applied. Typically, a wrapper method is a stepwise selection method. It is a greedy search that determines the best next selection step to make based on the current situation, by performing either a forward selection or backward selection. In forward selection method, we first build n separate one-dimension (d) models and choose the best one resulted from the measurement performed. Then we select that particular variable from that model and build n-1 2-d models using this first variable along with all other possibles. We select the best 2-d model and continue the same pattern until no more substantial model improvement. Backward selection methods illustrate the same logic but instead runs in the opposite direction. We first build a single model using all variables. Next, we build n models each removing one variable. Then we select the best model; it has n-1 variables. Then we build n-1 separate models each removing one variable and select the best model. This iterative pattern continues until the model degradation is below an acceptable amount. Eventually, a stepwise selection method results in an ordered list of variables sorted by the importance in predicting the dependent variable.

The main goal in this stage of machine learning problem is to find the small subset of informative variables and abandon large number of other variables. The wrapper involves a repeated, iterative process of building models, meaning many models have to be built to perform the simple task of variables selection. To be more effective in such a selecting process, wrapper feature selection process does not require the best model associate with it, and it is common to use a simple and fast linear model. This paper applies logistic regression model as wrapper and uses backward selection method.

The feature selection process begins with 308 expert variables. Univariate filter methods KS and FDR are first applied to remove all but 80 variables. Then we apply wrapper to reduce to 30 variables using backward selection with logistic regression model.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

6. MODEL ALGORITHMS

This paper uses four supervised machine learning models, starting with a baseline logistic regression model and a number of nonlinear models including decision tree, random forest, and artificial neural net (ANN). Data are separated into modeling period data (training, testing) and validation period data (out of time/OOT) before inputting into models. Validation data are data in a different point in time from our modeling data. We choose first 10 month of the dataset as our modeling data where we further randomly divide into training and testing using cross validation, bootstrapping, and bagging. Once the best model is achieved, it evaluates the data on the remaining 2-month data that is out of time.

Figure 7. The separation of data sets

The out of time performance is a good indication of how well our models perform on data that is never seen before. Typically, one should always work hard to maximize OOT performance with minimal model complexity. The OOT performance indicates what we can expect when implementing the model as it provides more realistic evaluations on unseen data.

6.1. Logistic Regression

Logistic regression is an appropriate regression analysis to use when the dependent variable is dichotomous in nature (fraud = 1, non-fraud = 0), to predict the probability of occurrence of a binary outcome. There are three basic steps in building logistic regression. The first step is to build the prediction function; the second step is to construct the cost function, and the third step is to minimize the cost function to improve the accuracy of the model. To address a classification problem, we use sigmoid/logistic activation function as the prediction function to map the predicted value to probability between [0,1] interval, transforming the output to a probability value that can be mapped to two discrete classes.

The sigmoid function defined as:

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1}. \tag{3}$$

e = base of natural log

x = input to the function (linear regression)

S(z) = probability output value between 0 and 1

Logistic regression is a special case of linear regression as it predicts the probabilities of outcome using log function. At the center of the logistic regression is the task estimating the log odds of an event. Log function transforms the prediction function into the cost function such that we can minimize the cost function later on. The final step uses gradient boost to locate the minimum of the cost function.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

Mathematically, logistic regression estimates a multiple linear regression function defined as:

$$a = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \quad (4)$$

b0 is the log-odds of the event that dependent variable Y = 1 (namely, fraud_label = 1), when the predictors xi = 0. With 30 candidate variables selected, logistic regression functions as a baseline model due to its simplicity of parameters interpretation.

In this model, three parameters are tuned to ensure the optimized performance:

penalty: penalized logistic regression imposes a penalty (regularization) to the model and shrinks the coefficients of the less contributive variables toward zero. This paper uses L1 norm regularization (aka. Lasso Regression) which adds "absolute value of magnitude" of coefficient as penalty term to the cost function.

C: inverse regularization parameter. It is the control variable that retains strength modification of regularization by being inversely positioned to the regularization strength (Lambda) regulator: C = 1/lambda. Higher values of C correspond to less regularization. C = .01 in this model.

n_jobs: used to specify the maximum number of concurrently running processor(s). n_jobs = -1 in this model, meaning all CPUs are used.

Table below shows the results of FDR on training, testing, and out-of-time data in 3 iterations:

Table 3. Results of Logistic Regression model

iteration	penalty	C	n_jobs	FDR_Train	FDR_Test	FDR_OOT
1	L1	0.01	-1	0.521429	0.517451	0.493713
2	L1	0.01	-1	0.520292	0.521469	0.494971
3	L1	0.01	-1	0.521494	0.513750	0.494971

6.2. Boosted Tree

Boosted Tree is an extension of the Decision Tree that have the output as a linear combination of a bunch of simple trees. It is a nonlinear supervised machine learning algorithm that trains a series of weak learners to result in a strong learner. Each weak learner is trained to predict the residual error of the current sum. When building the next tree, it looks through all the records from the data. Some of records were predicted well so the weight on those records are small; some of them were predicted poorly so the weight on those records for the next training iteration is high. So the next training iteration focus on those records that have been poorly predicted since they have higher weight, and ignores the records that were predicted well. Such a boosting process is an iterative approximation process where we incrementally add more trees, in a similar fashion to a Taylor series. Each addition should render more predictive power to the whole tree.

This paper uses xgboost library that provides a efficient boosted tree model named XGBClassifier. In this model, there are three major parameters adjusted:

n_estimators: the number of simple decision trees in the boosted tree model.

learning_rate: learning speed. The prediction accuracy might be low if learning rate is too large while the train time would be long if it is too small.

max_depth: the maximum depth that is allowed in a tree. it should not be too large to prevent overfitting.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

Model building result is below:

Table 4. Results of Boosted Tree model

iteration	n_estimator	Learning_rate	max_depth	FDR_Train	FDR_Test	FDR_OOT
1	1000	0.01	5	0.544222	0.545568	0.516065
2	500	0.01	6	0.546378	0.536439	0.515926
3	600	0.1	5	0.544656	0.555612	0.516065

6.3. Random Forest

A random forest is another improved version of a decision tree, and it is a nonlinear supervised machine learning algorithm that is an ensemble of many strong decision trees. It is a typical example of Ensemble / Committee Models (where we create many separate independent models that each predict the output). A random forest builds each decision tree by randomly selecting subsets of features in the base and each split iteration. The final output is the average (if a regression problem) or vote (if a classification problem) across all simple decision trees, thus a random forest improves the stability, robustness, and are less prone to overfit comparing to a complex decision tree.

In the training process, 5 major parameters are tuned, and these parameters are:

n_estimator: the number of trees. The increase in trees improves the accuracy of the prediction, however, costs more time. In addition, the improvement becomes subtle when the number of trees is unnecessarily large.

max_features: the number of features that are allowed in a tree. Allowing more features increases the performance and decreases the speed.

max_depth: the maximum depth that is allowed in a tree. The tree terminates when the path to the deepest leaf node exceeds the parameter.

min_samples_split: the minimum number of transactions required in the current node to split.

min_samples_leaf: the minimum number of transactions required at the current node. A larger parameter decreases the influence of noise in the data.

max_depth, min_samples_split, and min_samples_leaf decides the size of each tree in the forest. The balance between speed and performance is considered when choosing the parameters. Table below shows some parameters that have been tested and the final parameters that achieved the best performance.

6.4. Artificial Neural Network (ANN)

Neural Net is a nonlinear supervised machine learning algorithm that simulates human brain's neural network structure. A typical neural net consists of an input layer, some number of hidden layers and an output layer. All the independent variables (x's) form the input layer; the dependent variable y is the output layer; the hidden layer is a set of nodes or "neurons." Each node in the hidden layer receives weighted signals from all the nodes in the previous layer and does a transform on this linear combination of signals, and the activation functions can be sigmoid / logistic, relu and maxout functions. Sigmoid is the most common function to use while relu is widely used in deep Neural Net since it can prevent vanishing gradient. Maxout is a special activation function as it does not have a specific expression and it will change during the training process.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

Table 5. Results of Random Forest model

Best Parameters							
Batch Size	Hidden Layers	Nodes			Activation Function	Initial Learning Rate	
3000	2	(30,30)			Maxout	0.01	
	n_estimator	max_feature	max_depth	min_samples_split	Min_sample_leaf	Training FDR	OOT FDR
Group 1	100	15	80	60	5	55.21%	52.66%
Group 2	100	15	80	80	10	54.95%	53.67%
Group 3	100	9	80	80	10	54.84%	53.68%
Group 4	100	15	80	80	5	54.85%	53.97%
Group 5	100	9	80	80	10	54.82%	53.74%
Group 6	100	9	80	50	5	54.88%	53.92%

Neural Net is trained through a process called backpropagation, which involves comparing the output a network produces with the output it was meant to produce and using the difference between them to modify the weights and bias of the connections between the units in the network. This backpropagating process performs in a backward direction: output layer — hidden layer(s) — input layer. It intends to reduce the difference between actual and intended output to the point where the two exactly coincide. In this case, the outputs are two numbers add up to 1 which represent the probability of fraud and non-fraud of the input record.

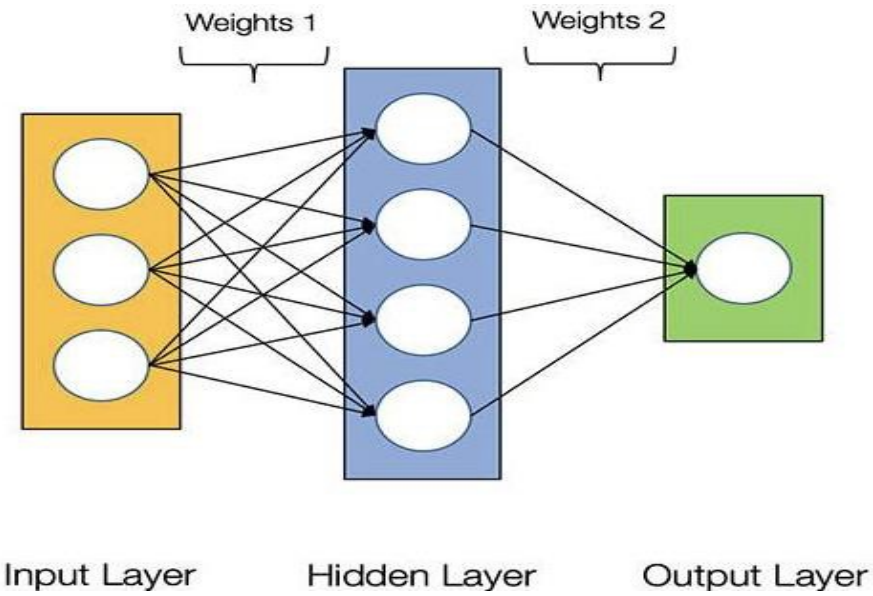

Figure 8. Structure of simple neural network

The basic parameters of the models are:

Batch Size: The number of samples used to update all the parameters of each nodes one time.

Hidden Layer(s): The number of hidden layers.

Nodes: The number of nodes in each layer.

Initial learning rate: The adam optimizer use adaptative learning rate, and it needs a learning rate to get started. The default is 0.001.

Activation function: Before each node outputs a number, a function will be applied to it in order to the network is not a linear combination. The sigmoid function is widely used but when the network goes deeper, the gradient vanishing will be a problem for this function. Relu is a function that can prevent this function. Maxout is an advanced version of relu. It does not use a specific activation function, but the function will be updated in each step.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

Model results on the card transaction data on the training, testing and out of time validation data sets. These numbers are the averages over 4 runs for each data set. The Epoch is 30 and the optimizer is Adam.

Some of the parameters with good performances are shown below.

Table 6. Results of ANN model

Best Parameters							
Batch Size		Hidden Layers		Nodes	Activation Function		Initial Learning Rate
3000		2		(30,30)	Maxout		0.01
BATCH_SIZE	Hidden_Layer(s)	Nodes	ACTIVATION_FUNC	INITIAL_Leaning RATE	0.555798	TEST(3%FDR)	OOT(3%FDR)
3000	1	15	Maxout(Maximum of two nodes)	0.001	0.554792	0.55	0.531643
3000	1	20	Maxout(Maximum of two nodes)	0.001	0.556384	0.550573	0.532272
3000	1	15	Sigmod	0.01	0.556831	0.551064	0.534577
3000	1	20	Sigmod	0.01	0.558032	0.551555	0.535624
3000	2	(20,20)	Maxout(Maximum of two nodes)	0.001	0.558451	0.550655	0.536253
3000	2	(30,30)	Maxout(Maximum of two nodes)	0.001	0.558423	0.550982	0.536987
5000	2	(30,30)	Maxout(Maximum of two nodes)	0.001	0.558479	0.550818	0.536148
3000	2	(30,30)	Maxout(Maximum of two nodes)	0.01	0.558311	0.551637	0.537091
5000	2	(20,20)	Maxout(Maximum of two nodes)	0.01	0.557473	0.551964	0.534786

7. FINAL MODEL

Having trained and evaluated the performance of logistic regression, boosted trees, a random forest, and neural network, we have found that the neural network has the best performance of identifying fraud. The neural network model outperformed all other models. Its FDR (fraud detection rates) at 3% cutoff for both the testing dataset and the out-of-time validation dataset are higher than all other models' results.

During the process of tuning Neural Network's parameters, we searched through a grid of 5 x 5 parameters, including batch size, number of hidden layers, number of nodes, activation function, and the learning rate. The combinations of these parameters that we tested out are partly listed in the section above. The exhaustive list is in the appendix. It turns out that the best parameter combination for Neural Network with the highest OOT FDR at 3% cutoff is:

batch size: 3000

layers: 2

nodes: (20,20)

activation_fun: maxout(max of two nodes outputs)

initial learning rate: 0.01

After getting these best parameters, we ran our model and calculated the FDR at 3% for training, testing, and out-of-time datasets. The results are:

training: 55.8311%

testing: 55.1637%

OOT: 53.7091%

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

The three tables show the final model performance statistics of the three data sets. And the OOT data set shows our best prediction of how the model will perform when applied to real-time data. In this table, we see that the model keeps most of the fraud records to the top bins. The bin statistics show the distribution of the data in each population bin and the cumulative statistics show how the model performs if any particular percentage is selected as a cutoff. The FPR (False Positive Rate) here is obtained by the number of cumulative goods divided by the number cumulative bads. A score cutoff of k% means that the model will regard the top k% of transaction as fraud sorted by the fraud score. Note that the FDR at 3% for each data set here are different from the table in model part. The table here is the final model runs on each data set one time while the tables in model part are the average FDR for over 4 runs in each data set.

7.1. Training Dataset Statistics

Table 7. Statistical data of final (ANN) model on training set

Training	Records#	Goods#	Bads#	Fraud Rate
	625130	616178	8952	0.01427703

Bin Statistics						Cumulative Statistics						
Population bin%	#records	#Goods	#Bads	Goods%	Bads%	Total Records#	Cumulative Goods#	Cumulative Bads#	%Goods	%Bads(FDR)	KS	FPR
1	6251	1556	4695	24.89201728	75.10798272	6251	1556	4695	0.25252433	52.44638 07	52.1938563	0.3314164
2	6251	6029	222	96.44856823	3.551431771	12502	7585	4917	1.231137756	54.92627346	53.6951357	1.5426072 8
3	6251	6172	79	98.73620221	1.263797792	18753	13757	4996	2.232796367	55.80875782	53.5759615	2.7536028 8
4	6251	6179	72	98.84818429	1.151815709	25004	19936	5068	3.235591014	56.61304736	53.3774563	3.9337016 6
5	6251	6210	41	99.34410494	0.665895057	31255	26146	5109	4.243416675	57.07104558	52.8276289	5.1176355 5
6	6251	6202	49	99.21612542	0.78387458	37506	32348	5158	5.250106 3	57.61840929	52.368303	6.2714230 3
7	6251	6214	37	99.4080947	0.591905295	43757	38562	5195	6.258581124	58.03172475	51.7731436	7.4229066 4
8	6251	6204	47	99.2481203	0.751879699	50008	44766	5242	7.265433 04	58.5567471	51.2913141	8.5398702 8
9	6251	6198	53	99.15213566	0.847864342	56259	50964	5295	8.271473503	59.14879357	50.8773201	9.6249291 8
10	6251	6209	42	99.3281075	0.671892497	62510	57173	5337	9.279136873	59.61796247	50.3388256	10.712572 6
11	6251	6187	64	98.97616381	1.023836186	68761	63360	5401	10.28322985	60.33288651	50.0449988	11.731160 9
12	6251	6206	45	99.28011518	0.719884818	75012	69566	5446	11.29056863	60.83556747	49.5449988	12.773778 9
13	6251	6202	49	99.21612542	0.78387458	81263	75768	5495	12.29709597	61.38293119	49.0858352	13.788535
14	6251	6208	43	99.31211006	0.687889938	87514	81976	5538	13.30459705	61.86327078	48.5586737	14.802455 8
15	6251	6219	32	99.48808191	0.511918093	93765	88195	5570	14.31404562	62.2207328	47.9066872	15.833931 8
16	6251	6198	53	99.15213566	0.847864342	100016	94393	5623	15.31992379	62.81277927	47.4928555	16.786946 5
17	6251	6209	42	99.3281075	0.671892497	106267	100602	5665	16.32758716	63.28194817	46.3592016	18.733076 1
18	6251	6214	37	99.4080947	0.591905295	112518	106816	5702	17.33606198	63.69526363	46.3592016	18.687859 3
19	6251	6212	39	99.37609982	0.623900176	118769	113028	5741	18.34437452	64.13092046	45.7865459	19.687859 3
20	6251	6202	49	99.21612542	0.78387458	125020	119230	5790	19.35090185	64.67828418	45.3273823	20.592400 7

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

7.2. Testing Dataset Statistics

Table 8. Statistical data of final (ANN) model on Testing set

Testing	Records#	Goods#	Bads#	Fraud Rate
	208377	20532	3055	0.014660927

Bin Statistics						Cumulative Statistics						
Population bin%	#records	#Goods	#Bads	Goods %	Bads%	Total Records#	Cumulative Goods#	Cumulative Bads#	%Goods	%Bads(FDR)	KS	FPR
1	2083	510	1573	24.48391743	75.51608257	2083	510	1573	0.248877373	51.4893617	51.2404843	0.32422123
2	2083	2002	81	96.11137782	3.88862218	4166	2512	1654	1.224418231	54.14075286	52.9163346	1.51874244
3	2083	2053	30	98.55976956	1.440230437	6249	4565	1684	2.224311082	55.12274959	52.8984385	2.7108076
4	2083	2064	19	99.08785406	0.912145943	8332	6629	1703	3.229561372	55.77741408	52.5478527	3.89254257
5	2083	2070	13	99.37590014	0.624099856	10415	8699	1716	4.238707981	56.17021277	51.9315048	5.06934732
6	2083	2065	18	99.13586174	0.864138262	12498	10764	1734	5.244933235	56.7594108	51.5144785	6.20761246
7	2083	2069	14	99.32789246	0.672107537	14581	12833	1748	6.252617839	57.21767594	50.9650581	7.34153318
8	2083	2058	25	98.79980797	1.200192031	16664	14891	1773	7.254954589	58.06873977	50.8137939	8.39875917
9	2083	2062	21	98.99183869	1.008161306	18747	16953	1794	8.260119618	58.72340426	50.4632081	9.44983278
10	2083	2077	6	99.71195391	0.288046087	20830	19030	1800	9.272265028	58.9198036	49.6475386	10.5722222
11	2083	2063	20	99.03984638	0.960153625	22913	21093	1820	10.27702828	59.57446809	49.2974398	11.5895604
12	2083	2067	16	99.2318771	0.768229	24996	23160	1836	11.28422673	60.09819967	48.8139729	12.6143791
13	2083	2068	15	99.27988478	0.720115218	27079	25228	1851	12.29191222	60.58919804	48.2972858	13.6293895
14	2083	2075	8	99.61593855	0.38406145	29162	27303	1859	13.30300698	60.85106383	47.5480568	14.6869285
15	2083	2069	14	99.32789246	0.672107537	31245	29372	1873	14.31117951	61.30932897	46.9981495	15.6817939
16	2083	2065	18	99.13586174	0.864138262	33328	31437	1891	15.31691684	61.898527	46.5816102	16.6245373
17	2083	2068	15	99.27988478	0.720115218	35411	33505	1906	16.32460233	62.3892537	46.064923	17.5786988
18	2083	2070	13	99.37590014	0.624099856	37494	35575	1919	17.3332619	62.8150728	45.4817954	18.5383012
19	2083	2071	12	99.42390783	0.576092175	39577	37646	1931	18.34240851	63.2078597	44.8654475	19.4955981
20	2083	2074	9	99.56793087	0.432069131	41660	39720	1940	19.3525292	63.5024499	44.1499258	20.4742268

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

7.3. OOT Dataset Statistics

Table 9. Statistical data of final (ANN) model on OOT set

OOT	Records#	Goods#	Bads#	Fraud Rate
	166493	164107	2396	0.014330933

Bin Statistics						Cumulative Statistics						
Population bin%	#records	#Goods	#Bads	Goods%	Bads%	Total Records#	Cumulative Goods#	Cumulative Bads#	%Goods	%Bads(FDR)	KS	FPR
1	1664	476	1188	28.605769923	71.394230077	1664	476	1188	0.290664018	49.790444426	49.499780200067	0.4006734
2	1664	1594	70	95.793269923	4.206730769	3328	2070	1258	1.262590871	52.724202464	51.461633801646	1.645469
3	1664	1649	15	99.098557692	0.901442308	4992	3719	1273	2.268032442	53.252891870848	51.084859402144	2.9214454
4	1664	1642	22	98.677884623	1.322115385	6656	5361	1295	3.269208504	54.274937130057	51.005728613976	4.139768340
5	1664	1645	19	98.858173081	1.141826923	8320	7006	1314	4.272212642	55.071248950799	50.799036305181	5.331811260
6	1664	1654	10	99.399038460	0.600961538	9984	8660	1324	5.280701006	55.490360445096	50.209659406078	6.5407855
7	1664	1656	8	99.519230773	0.480769231	11648	10316	1332	6.290408087	55.825649624961	49.535241574747	7.7447474
8	1664	1644	20	98.798076923	1.201923077	13312	11960	1352	7.292193508	56.663872597916	49.371679138461	8.8461538
9	1664	1650	14	99.158653853	0.841346154	14976	13610	1366	8.298244438	57.250628674823	48.952384296633	9.9633967
10	1664	1653	11	99.338942307	0.661057692	16640	15263	1377	9.305514085	57.753556245048	48.448048416411	11.084241
11	1664	1652	12	99.278846153	0.721153846	18304	16915	1389	10.313393093	58.214585085082	47.901192125825	12.177825
12	1664	1646	18	98.918269230	1.081730769	19968	18561	1407	11.317006596	58.968985754792	47.651979215977	13.191897
13	1664	1652	12	99.278846153	0.721153846	21632	20213	1419	12.324276231	59.471919534733	47.147643303384	14.244538
14	1664	1647	17	98.978365384	1.021634615	23296	21860	1436	13.328499099	60.184409050865	46.855911	15.222841
15	1664	1650	14	99.158653853	0.841346154	24960	23510	1450	14.334550026	60.771165136513	46.436615131	16.213793
16	1664	1652	12	99.278846153	0.721153846	26624	25162	1462	15.341819671	61.274098910792	45.932279239	17.210670
17	1664	1656	8	99.519230773	0.480769231	28288	26818	1470	16.351526750	61.609388145.257861374	18.243537
18	1664	1653	11	99.338942307	0.661057692	29952	28471	1481	17.359405750	62.070410730544.711005	19.224172
19	1664	1653	11	99.338942307	0.661057692	31616	30124	1492	18.367284760	62.531433360044.164148620.190348	5
20	1664	1655	9	99.459134620	0.540865385	33280	31779	1501	19.376382480	62.908633743.532251221.171885	4

8. CONCLUSION

This paper explores the application of linear and nonlinear supervised machine learning models on credit card transaction data, to identify fraudulent credit card transactions and behaviors. The result indicates that all nonlinear models slightly outperform the linear model, and neural net is determined to be the best predictive model with out-of-time FDR of 53.7091%. More variable encoding or data fields (e.g. time of day, or supplementary cardholder information) can certainly help improve model performance. Further model parameter tuning can also provide improvements to the models. The resulting model can be utilized in a credit card fraud detection system, and the similar model development process can be performed in related business domains such as insurance and telecommunications, to avoid or detect fraudulent activity.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0033

REFERENCES

[1] Credit card debt statistics for 2019. (n.d.) The Ascent. https://www.fool.com/the-ascent/research/credit-card-debt-statistics/

[2] Hastie, T., Tibshirani, R. and Friedman, J. (2009) Boosting and Additive Trees. In: The Elements of Statistical Learning, Springer, New York, 337-387.

[3] Jensen, D. (1997) Prospective Assessment of AI Technologies for Fraud Detection: A Case Study. In: The AAAI Workshop on AI Approaches to Fraud Detection and Risk Management, AAAI Press, Palo Alto, CA, 34-38.

[4] Randhawa, K., Loo, C.K., Seera, M., Lim, C.P. and Nandi, A.K. (2018) Credit Card Fraud Detection Using AdaBoost and Majority Voting. IEEE Access, 6, 14277-14284. https://doi.org/10.1109/ACCESS.2018.2806420

[5] Ryan, J., Lin, M.-J. and Miikkulainen, R. (1998) Intrusion Detection with Neural Networks. In: Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 943-949.

[6] Green, B.P. and Choi, J.H. (1997) Assessing the Risk of Management Fraud through Neural Network Technology. Auditing, 16, 14-28.

[7] Gao, J.X., Zhou, Z.R., Ai, J.S., Xia, B.X. and Coggeshall, S. (2019) Predicting Credit Card Transaction Fraud Using Machine Learning Algorithms. Journal of Intelligent Learning Systems and Applications, 11, 33-63. https://doi.org/10.4236/jilsa.2019.113003

[8] Albashrawi, M. (2016) Detecting Financial Fraud Using Data Mining Techniques: A Decade Review from 2004 to 2015. Journal of Data Science, 14, 553-569.

[9] Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S. and Bontempi, G. (2014) Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective. Expert Systems with Applications, 41, 4915-4928. https://doi.org/10.1016/j.eswa.2014.02.026

[10] Estévez, P.A., Held, C.M. and Perez, C.A. (2006) Subscription Fraud Prevention in Telecommunications Using Fuzzy Rules and Neural Networks. Expert Systems with Applications, 31, 337-344. https://doi.org/10.1016/j.eswa.2005.09.028

[11] Navneet Jain, V.K. (2018) Credit Card Fraud Detection Using Recurrent Attributes. International Advanced Research Journal in Science, Engineering and Technology, 5, 43-47.

[12] Wheeler, R. and Aitken, S. (2000) Multiple Algorithms for Fraud Detection. In: Ellis, R., Moulton, M. and Coenen, F., Eds., Applications and Innovations in Intelligent Systems VII, Springer, London, 219-231. https://doi.org/10.1007/978-1-4471-0465-0_14

[13] Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S. and Jiang, C. (2018) Random Forest for Credit Card Fraud Detection. 2018 IEEE 15th International Conference on Networking, Sensing and Control, Zhuhai, 27-29 March 2018, 1-6. https://doi.org/10.1109/ICNSC.2018.8361343

[14] Q. Wang, X. Li and Q. Qin, "Feature Selection for Time Series Modeling," Journal of Intelligent Learning Systems and Applications, Vol. 5 No. 3, 2013, pp. 152-164.