Fraud detection in mobile money transfer as binary classification problem

Ratha Pech

ratha.pech@gmail.com

Abstract. Electronic money transfer through mobile or device has been increasingly grown in the last few years with the boom of smart phones. This convenience of transferring and receiving money has attracted not only the users, but also the attackers and fraudsters. The traditional rule-based methods require much time and labor to detect fraudulent activities. With large amount of data collecting everyday, rule-based method may not be capable of dealing with this scenario on time. Hence, adaptive computational methods, e.g., machine learning and data mining, play important role to assist experts to identify such illegal activities by minimizing false alarms and increasing true positive frauds. In present study, we discuss a fraud detection problem in mobile money transfer as binary classification, then machine learning methods have been applied to identify whether the transactions are fraud or not. The simulation results shows that the machine learning methods are the critical tools to help solving this problem.

Keywords: fraud detection, binary classification, support vector machine, multilayer perceptron, Naïve Bayes classifier

1 Introduction

In the last few years, mobile money transfer which is the process of transferring money from one mobile or other device to another using digital equivalent of cash, i.e., electronic money, without involving any bank accounts has been exponentially growing through the emergence of smart phones and online service. Mobile money transfer simplifies banking relationships and facilitates financial inclusion, therefore, it is rapidly expanding especially in developing countries. Mobile money transferring systems are subject to the same controls as those required for financial institutions, including the detection of money laundering and other illegal money transferring. The growing of mobile money transfer becomes the target of attackers and fraudsters. Hence, intelligent computational techniques such as machine learning and data mining [1,2,3,4] should be the critical tools for helping detecting such illegal activities.

One of the limitation of the research in this kind of topic is the lack of public data for a sound comparison among any methods. Many literature used their private data which are subjected to privacy and confidentiality and cannot be disclosed because of vulnerabilities of services and obligation to maintain

privacy of the companies' clients. That is quite understandable. Thank to PaySim dataset, a simulated database have been generated for the purpose of research [5].

In this report, we discuss the fraud detection problem as binary classification by using three machine learning methods namely support vector machine (SVM), multilayer perceptron (MLP) and Naïve Bayes (NB) to solve this problem based on a public PaySim dataset. Specifically, given enough training data, e.g., including both fraud transactions and non-fraud transactions, the objective of binary classification is to train the classifiers from historical data by using machine learning methods, so that they are capable of identifying whether new transactions are frauds or non-frauds. We discuss from data preprocessing, sampling to feeding the model, then we evaluate the methods and report their performances.

2 Related works

Financial frauds including credit card fraud, corporate fraud and money laundering have increasingly attracted concern and attention [6]. Data mining which aims at uncovering the hidden patterns in the data is one the popular techniques for detecting fraud. Data mining adopts various methods from statistical learning, mathematics, machine learning and artificial intelligence to extract these hidden patterns inside the data. Neural network is a traditional machine learning method which is inspired by human brain computation [7]. It is one of the powerful machine learning methods in solving fraud detection problem [8,9,10]. In [8], Dorronsoro *et al* have built online system to classify the fraud of credit card operations based on a neural classifier. The authors claimed that in their problem, neural network performs satisfactorily. Artificial neural network has also been used with standard statistical tools to detect fraud in financial statements in [9]. The method has provided empirical evidence of red flag suggestion. Moreover, in [10] Viaene *et al* have proposed neural network classifiers with automatic relevance determination weight regularization for fraud detection in personal injury protection automobile insurance claimation.

Naïve Bayes classifier has been utilized in [11] with asymmetric or skewed logit link to fit a fraud database from the Spanish insurance market. In that study, the authors regarded the fraud detection problem as binary classification. Moreover, Viaene *et al* in [12] have used AdaBoosted Naïve Bayes to diagnose insurance claim fraud. The empirical results based on accidents that occurred in Massachusetts (USA) during 1993 shew the effectiveness of the method.

Support vector machine (SVM) and quarter-sphere support vector machine [13] have been implemented to detect fraud in mobile telecommunication networks by comparing the most recent call patterns of users with their past usage patterns. The methods have been shown to perform satisfactorily on detecting fraudulent call without raising too many false positive errors. Support vector machine and decision tree have been compared on credit card fraud detection

problem based on the real dataset [14]. SVM has also been used in [15] to detect fraud in the credit card usage based on users' behaviors.

Our study share some commons with those in [16] in which authors have proposed the approach to detect the fraud in mobile payment system. However, we more focus on the fraud detection as binary classification problem and we compare the performances of three methods in the machine learning on this problem. In addition, we test the models with the public dataset which can be used in either commercial application or academic research. For further discussions on fraud detection problem, readers may consult a great number of survey which have been conducted on this problem [3,6,17,18,19].

3 Data

Fig. 1 illustrates the interrelated stages denoted as arrows from collecting data to preprocessing, training and validating model. In other words, when the trained model is shown to perform unsatisfactorily, the earlier stages are needed to be redesigned. The data obviously are very critical in the whole process since the performance of the model heavily depends on them. If the data are not properly dealt, the solutions as well as results can mislead the conclusion and understanding finally leading to wrong decision making. Because the public data of fraud detection are limited, thanks to PaySim dataset [1] which is computationally simulated based on real data [5], we adopt this dataset to discuss in this report. The PaySim dataset contains eleven columns such as:

- step: maps a unit of time in the real world. In this case 1 step is 1 hour of time
- type: CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER
- amount: amount of the transaction in local currency
- nameOrig: customer who started the transaction
- oldbalanceOrg: initial balance before the transaction
- newbalanceOrig: customer's balance after the transaction
- nameDest: recipient ID of the transaction
- oldbalanceDest: initial recipient balance before the transaction
- newbalanceDest: recipient's balance after the transaction
- isFraud: identifies a fraudulent transaction (1) and non-fraudulent (0)
- isFlaggedFraud: flags illegal attempts to transfer more than 200.000 in a single transaction.

3.1 Preprocessing

Collected data are normally cannot be immediately feed into the machine learning algorithms. They contain irrelevant fields, noise, missing values and sometimes they are highly unstructured. Therefore, before feeding them to the model

[1] https://www.kaggle.com/ntnu-testimon/paysim1

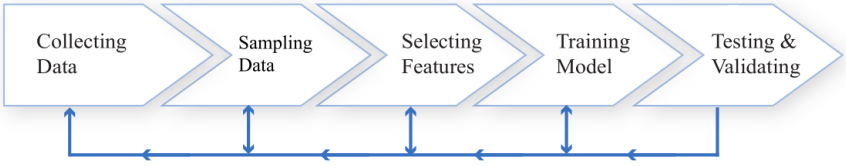

Fig. 1. A generic framework of mobile money transfer fraud detection process

to train, we need to preprocess such as denoising, randomly sampling, selecting informative features and dealing with missing values and structuring. Fortunately, PaySim data are structured and contain no missing values.

Among total records which is more than 1 million records, there only 1142 records are labeled as frauds. Therefore, the distributions of these two classes are highly skewed. In order to mitigate this problem, we adopt a statistical technique namely random sampling. The motivation that we use this technique is that in real world problem the data that are collected are normally unordered. We use all the 1142 records of fraud transactions as positive samples, then we randomly select from non-fraud transactions as negative samples. In order to find the good data representatives, we independently and randomly select a subset of non-fraud transactions. Then validate the performance of the methods on these subsets one by one.

The number of features of PaySim data is quite small, e.g., eleven columns. Hence, we do not need complicated techniques to process feature selection. We simply remove three columns such as nameOrig, nameDest and isFlaggedFraud. Hence, we keep only eight columns including step, type, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest and isFraud to train the model. isFraud is used as labels of the records. The column type contains categorical values which are CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER. We need to transfer this column into dummy variables such that the column type is transferred into four columns as defined in table 1. Moreover, we standardize the data by using scaling method to make the data more like standard normally distributed data, i.e., Gaussian with zero mean and unit variance.

Table 1. The categorical values of column type are converted into dummy variables

type	dummy 1	dummy 2	dummy 3	dummy 4
CASH-IN	0	0	0	1
CASH-OUT	0	0	1	0
DEBIT	0	1	0	0
PAYMENT	1	0	0	0
TRANSFER	1	1	1	1

4 Methods

Binary classification problem is a special case of multi-class classification problem where there are only two classes of all the data utilized in the model. Fraud detection problem on one hand can be regarded as binary classification such that transactions is classified as fraud or non-fraud. On the other hand, fraud detection can also be considered as one-class classification [22] or outlier detection where the fraud transactions are regarded as outliers since they are different from normal transactions. In this study, however, we discuss this problem as binary classification problem and leave the second scenario to the future work. We adopt three popular machine learning methods namely support vector machine (SVM), multi-layer perceptron (MLP) and Naïve Bayes (NB) to solve this problem. In the following sections, we discuss these methods in a bit more details.

Notations
We illustrate the mathematical notations in this report as follows.

- $D = D_1, D_2, ..., D_n$ is the set of n examples
- $\mathbf{x}_i = x_{1,i}, x_{i,2}, ..., x_{i,d}$ is an input vector with d dimension or features
- y_i is the known label of each example and given by expert or teacher
- $D_i =< \mathbf{x}_i, y_i >$ is the pair of $\mathbf{x}_i$ and its label y_i
- $\mathbf{w}$ is the weight vectors

The objective of binary classification is to train a mapping model $f : X \longrightarrow Y$ subject to $y_i \approx f(\mathbf{x}_i)$ for all $x = 1, 2, ..., n$. In this study, y_i can be 1 (fraud) and 0 (non-fraud). The objective function $f(\mathbf{x}_i)$ can have different forms according to the training models. For instance, $f(\mathbf{x}_i)$ is the probability score in Naïve Bayes classifier.

4.1 Support vector machine

Support vector machine (SVM) [25] for binary classification aims at computing the hyperplane or a set of hyperplanes in high dimensional feature space that can separate data points between the two classes in that space by maximizing the margin between the classes' closest points. Whenever new data come, the classifier will identify their classes based on the side of the hyperplane they fall.

Given a training dataset of n points in the form of $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$, where y_i is the label that $\mathbf{x}_i$ belong to and y_i can be 1 or -1 (in our case, non-fraud is denoted as 0, but it can be easily converted to -1 for mathematical convenience). When the data points can be linearly separated, the hyperplane can take the form as

$$\begin{aligned} \mathbf{w}\mathbf{x}_i + b \geq 1 \quad \text{if} \quad y_i = 1, \quad \text{or} \\ \mathbf{w}\mathbf{x}_i + b \leq -1 \quad \text{if} \quad y_i = -1. \end{aligned} \tag{1}$$

The Eq. (1) can be rewritten as

$$y_i(\mathbf{w}\mathbf{x}_i + b) \geq 1, \quad i = 1, 2, ..., n. \tag{2}$$

However, when the data are linearly inseparable, kernel trick [23] has been adopted. Some common kernel functions include Polynomial, Gaussian radial basis and hyperbolic tangent. In this study, we adopt Gaussian radial basis function (RBF) [24] for the classification. RBF kernel on two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as

$$K = \exp\left(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2\right), \tag{3}$$

where γ is a free parameter. The radial basis function classifier is obtained as

$$f(\mathbf{x}) = \text{sgn}\left[\sum_{i=1}^{n} w_i \exp\left(-\gamma||\mathbf{x} - \mathbf{x}_i||^2\right) + b\right], \tag{4}$$

where b is a constant, $\mathbf{x}_i$ is the center of the circle and sgn is the sign of the function. SVM with RBF kernel aims at computing the location of centers $\mathbf{x}_i$, $\mathbf{w}$ and b that map the patterns nonlinearly in a high-dimensional feature space.

4.2 Multilayer perceptron

Artificial neural network, commonly known as neural network (NN), algorithm is inspired by human brain [7]. NN is one of the popular methods in machine learning that can solve many complicated problems such as pattern recognition, perception, information processing and fusion, etc. Multilayer perceptron (MLP), which is a type of NN, consists of at least three layers of neurons in which the first layer is the input, while the last layer is the output and the middle layer is the hidden layer. The main goal of MLP is to obtain the synaptic weights $\mathbf{w}$ such that it can map the input $\mathbf{x}_i$ (training data) to the output y_i (target label). The model of each neuron in the network includes a nonlinear activation function which is differentiable. The network can contain one or more layers. The network exhibits a high degree of connectivity, the extent of which is determined by synaptic weights of the network.

Backpropagation is used to train the MLP in which two phases are involved. The first phase is called forward. In particur, the synaptic weights of the network are fixed and the input signal is propagated through the network, layer by layer, until it reaches the output. Thus in this phase, changes are confined to the activation potentials and outputs of the neurons in the network. The three common activation functions include:

$$g = \sum_{i=1}^{n} \mathbf{w}_i\mathbf{x}_i + b, \quad g = \tanh\left(\sum_{i=1}^{n} \mathbf{w}_i\mathbf{x}_i + b\right) \quad \text{and} \quad g = \left(1 + \exp^{\left(-\sum_{i=1}^{n} \mathbf{w}_i\mathbf{x}_i + b\right)}\right)^{-1}. \tag{5}$$

Another phase is known as backward where an error is produced by comparing the output of the network with a desired target. The result error is propagated through the network, again layer by layer, but this time the propagation is performed in the backward direction. In this second phase, the successive adjustments are made to the synaptic weights of the network. The error in output

node j in the n^{th} data point denoted by $e_j(n)$ is obtained by

$$e_j(n) = y_j(n) - g_j(n), \tag{6}$$

where g_j is the output obtained from the activation function and y_j is the target output or label. The node weights are adjusted based on corrections that minimize the error in the entire output, given by

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n). \tag{7}$$

Using gradient descent, the change in each weight is

$$\delta w_{ji} = -\eta \frac{\partial \mathcal{E}(n)}{\partial x_j(n)} g_i(n), \tag{8}$$

where η is the learning rate which is selected to make the weights converge quickly to a response, without oscillations.

The activation functions of LMP can be linear or nonlinear. Some of the common activation functions include rectified linear unit, sigmoid and tangent function. In the present report, we adopt rectified linear units (ReLU) [26,27] as the activation function. The predicted class obtained from rectified linear unit is obtained as

$$f(\mathbf{x}) = \text{argmax max}\left(0, \sum_{i=1}^{n} \mathbf{w}_i \mathbf{x}_i + b\right). \tag{9}$$

4.3 Naïve Bayes

Naïve Bayes classifier, which is based on Bayes' theorem [20], is one of the popular machine learning methods since its mathematical computation is effective and efficient. Naïve Bayes is based on a strong assumption that features are independent. The posterior probability of Bayes theorem is computed as

$$P(y|\mathbf{x}) = \frac{P(y) \times P(\mathbf{x}|y)}{P(\mathbf{x})}, \tag{10}$$

where $P(y|\mathbf{x})$ is the posterior probability, $P(y)$ is the prior probability, $P(\mathbf{x}|y)$ is the likelihood and $P(\mathbf{x})$ is the evidence. $P(\mathbf{x})$ is the same for all classes, hence it does not affect the posterior probability, then we can remove this term. With the assumption that all input features are conditionally independent of each other given the class, Naïve Bayes classifier is defined as

$$P(\mathbf{x}, y) = P(y) \times \prod_{i=1}^{d} P(x_i|y). \tag{11}$$

The conditional probability $P(x_i|y)$ can have different models, e.g., the features can be modeled as Bernoulli, multinomial, Gaussian density or Poisson distribution. In this study, we adopt Gaussian Naïve Bayes since it works well with

continuous values of features. The conditional probability $P(x_i|y)$ with Gaussian Naïve Bayes given class y_k can be obtained as

$$P(x_i|y_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}} \tag{12}$$

where μ_k is the mean of x in associated with class y_k, σ_k^2 is the variance of the values in x associated with class y_k.

5 Simulations

The simulations are conducted on three popular methods in fraud detection problem including support vector machine (SVM) with RBF kernel trick, multi-layer perceptron (MLP) with rectified linear unit activation function and Gaussian Naïve Bayes (NB). Note that there is no hyper-parameter in NB. On the other hand, SVM contains some hyper-parameters such as γ as defined in Eq. (4), tolerant residual and/or maximum number of iteration to ensure the convergence of the model before it stops training. MLP also contains a few hyper-parameters such as number of hidden layer, number of neuron in each hidden layer, learning rate, momentum, tolerant residual and maximum number of iteration. We implement the three methods by using Python and scikit-learn library [21].

5.1 Data experimental setup

We adopt random process for selecting a subpopulation of existing data for training the models and testing the performances of the three methods. Since, there only 1142 records are labeled as frauds, we use all of them in each selection process. On the other hand, there are more than a million records which are labeled as non-frauds, hence we conduct a random process to select from 10000 up to 50000 from total non-fraud transactions with the increment of 10000 for each simulation. Among these selected data, we randomly divide them into training and testing set. Moreover, to test the effectiveness of the methods against limited information of the training data, especially the transactions with frauds, from fraud and non-fraud transactions we randomly and equally select 200 up to 1000 with the increment of 200 for testing data and leave the remaining transactions as training data. In addition, to test the stability of the three methods, we independently run the simulations 10 times for each data division and report their average values as well as standard errors.

5.2 Metrics

Four popular metrics e.g., accuracy, precision, recall, F-score are adopted to validate the performances of the methods. Accuracy is defined as the summation of number of true positive (T_p) and true negative (T_n) divided by total testing

records, i.e., summation of true positive, true negative, false positive (F_p) and false negative (F_n).

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}. \tag{13}$$

Precision is defined as the number of true positive divided by the summation of true positive and false positive. Mathematically, it is written as

$$Precision = \frac{T_p}{T_p + F_p}. \tag{14}$$

On the other hand, recall is defined as number of true positive divided by the summation of true positive and false negative. Alternatively, it is defined as

$$Recall = \frac{T_p}{T_p + F_n}. \tag{15}$$

F-score is defined as

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{16}$$

5.3 Results

We illustrate the performances of the three methods measured by accuracy, precision, recall and F-score in table 2, 3, 4 and 5, respectively. The results show that MLP slightly outperforms SVM in term of accuracy and precision, but it perform much better than NB. However, NB is more effective than MLP and SVM in term of recall. Note that recall does not consider false negative rates. In term of F-score, MLP performs slightly better than the other two methods. In overall, in term of effectiveness of the fraud detection methods, MLP and SVM is more effective than NB. Note also that in each data division, we keep the number of fraud and non-fraud transactions in the testing data the same in order to avoid majority and minority problem. For example, we have 5% of testing data are fraud and the remaining are frauds. If the algorithms classify all the testing records as non-fraud transactions, the precision should be 95% accurate. However, the results become meaningless.

When we increase the number of testing transactions, the performances of SVM measured by accuracy, recall and F-score drop quickly following by MLP and NB. That is because the positive samples (eg., the fraud transactions) become limited for SVM to train the hyperplanes. NB is quite robust even when the number of positive samples is limited. Therefore, in case that we have only a small number of positive samples, NB should be a good start.

6 Conclusion and future work

In this report, we discuss a financial fraud detection problem in mobile money transfer as supervised learning namely binary classification. We adopt three popular machine learning methods including support vector machine (SVM), multilayer perceptron (MLP) and Naïve Bayes (NB) to solve this problem. The

Table 2. The average accuracy values of the three methods with different sizes of testing data (e.g., 200, 400, 600, 800 and 10000 from each class) and different size of sample data. The values in the brackets are the standard errors.

Sample	Method	200	400	600	800	1000	Average
10000	SVM	0.9405 (0.0087)	0.9349 (0.0070)	0.9279 (0.0084)	0.9130 (0.0074)	0.8292 (0.0217)	0.9091 (0.0062)
	MLP	**0.9420** (0.0100)	**0.9414** (0.0070)	**0.9346** (0.0046)	**0.9240** (0.0101)	**0.8970** (0.0097)	**0.9278** (0.0187)
	NB	0.7990 (0.0139)	0.7809 (0.0103)	0.7879 (0.0038)	0.7834 (0.0094)	0.7955 (0.0078)	0.7893 (0.0078)
20000	SVM	0.9273 (0.0063)	0.9150 (0.0149)	0.9120 (0.0071)	0.8626 (0.0095	0.7615 (0.0157)	0.8757 (0.068)
	MLP	**0.9320** (0.0086)	**0.9263** (0.0095)	**0.9226** (0.0070)	**0.9078** (0.0156)	**0.8787** (0.0159)	**0.9135** (0.0214)
	NB	0.7883 (0.0131)	0.7836 (0.0134)	0.7813 (0.0110)	0.7826 (0.0098)	0.7960 (0.0051)	0.7863 (0.0060)
30000	SVM	**0.9308** (0.0072)	0.9198 (0.0076)	0.8997 (0.0138)	0.8677 (0.0202)	0.7284 (0.0190)	0.8692 (0.0823)
	MLP	0.9303 (0.0125)	**0.9216** (0.0092)	**0.9167** (0.0079)	**0.8992** (0.0096)	**0.8573** (0.0144)	**0.9050** (0.0290)
	NB	0.7818 (0.0152)	0.7898 (0.0088)	0.7813 (0.0079)	0.7839 (0.0090)	0.7938 (0.0135)	0.7861 (0.0055)
40000	SVM	**0.9228** (0.0127)	**0.9184** (0.0077)	0.8960 (0.0174)	0.8414 (0.0239)	0.7209 (0.0219)	0.8599 (0.0842)
	MLP	0.9218 (0.0116)	0.9181 (0.0086)	**0.9089** (0.0093)	**0.8911** (0.0111)	**0.8507** (0.0121)	**0.8981** (0.0291)
	NB	0.7765 (0.0160)	0.7863 (0.0102)	0.7852 (0.0132)	0.7838 (0.0111)	0.7900 (0.0138)	0.7843 (0.0050)
50000	SVM	**0.9233** (0.0102)	**0.9175** (0.0095)	0.8961 (0.0122)	0.8358 (0.0218)	0.7259 (0.0194)	0.8597 (0.0824)
	MLP	0.9173 (0.0092)	0.9125 (0.0111)	**0.9029** (0.0092)	**0.8833** (0.0186)	**0.8520** (0.0139)	**0.8936** (0.0266)
	NB	0.7883 (0.0130)	0.7844 (0.0157)	0.7883 (0.0074)	0.7813 (0.0060)	0.7920 (0.0088)	0.7869 (0.0041)
Average	SVM	**0.9289** (0.0072)	0.9211 (0.0079)	0.9063 (0.0137)	0.8641 (0.0305)	0.7532 (0.0454)	0.8747 (0.0204)
	MLP	0.9287 (0.0096)	**0.9240** (0.0110)	**0.9171** (0.0123)	**0.9011** (0.0157)	**0.8671** (0.0201)	**0.9076** (0.0136)
	NB	0.7868 (0.0084)	0.7850 (0.0033)	0.7848 (0.0034)	0.7830 (0.0011)	0.7935 (0.0025)	0.7861 (0.0020)

Table 3. The average precision values of the three methods with different sizes of testing data (e.g., 200, 400, 600, 800 and 10000 from each class) and different size of sample data. The values in the brackets are the standard errors.

Sample	Method	200	400	600	800	1000	Average
10000	SVM	0.9858 (0.0085)	0.9863 (0.0036)	0.9888 (0.0031)	0.9886 (0.0049)	0.9967 (0.0023)	0.9892 (0.0044)
	MLP	**0.9950** (0.0046)	**0.9969** (0.0026)	**0.9969** (0.0021)	**0.9975** (0.0015)	**0.9987** (0.0010)	**0.9970** (0.0013)
	NB	0.7144 (0.0148)	0.6969 (0.0098)	0.7047 (0.0033)	0.7005 (0.0103)	0.7159 (0.0093)	0.7065 (0.0084)
20000	SVM	0.9971 (0.0038)	0.9946 (0.0028)	0.9945 (0.0035)	0.9981 (0.0017)	0.9972 (0.0020)	0.9963 (0.0017)
	MLP	**1.0000** (0.0000)	**0.9968** (0.0009)	**0.9975** (0.0018)	**0.9991** (0.0012)	**0.9997** (0.0005)	**0.9986** (0.0014)
	NB	0.7042 (0.0134)	0.7014 (0.0130)	0.6981 (0.0120)	0.7006 (0.0102)	0.7146 (0.0070)	0.7038 (0.0064)
30000	SVM	0.9909 (0.0058)	0.9927 (0.0046)	0.9967 (0.0010)	0.9972 (0.0017)	0.9978 (0.0018)	0.9951 (0.0031)
	MLP	**0.9977** (0.0028	**0.9985** (0.0020 (	**0.9980** 0.0016	**0.9986** (0.0015	**0.9993** (0.0007	**0.9984** (0.0006
	NB	0.6984 (0.0145)	0.7067 (0.0086)	0.6975 (0.0088)	0.7021 (0.0101)	0.7138 (0.0142)	0.7037 (0.0067)
40000	SVM	0.9959 (0.0046	0.9965 (0.0037)	0.9990 (0.0010)	0.9975 (0.0011)	0.9975 (0.0022)	0.9972 (0.0012)
	MLP	**0.9988** (0.0024)	**0.9991** (0.0014)	**0.9996** (0.0008)	**0.9995** (0.0007)	**0.9999** (0.0004)	**0.9994** (0.0004)
	NB	0.6927 (0.0151)	0.7047 (0.0100)	0.7033 (0.0141)	0.7029 (0.0130)	0.7086 (0.0156)	0.7024 (0.0059)
50000	SVM	0.9960 (0.0044)	0.9973 (0.0037)	0.9977 (0.0021)	0.9977 (0.0022)	0.9984 (0.0024)	0.9974 (0.0009)
	MLP	**0.9994** (0.0017)	**0.9997** (0.0009)	**0.9985** (0.0016)	**0.9995** (0.0008)	**0.9994** (0.0010)	**0.9993** (0.0004)
	NB	0.7042 (0.0131)	0.7009 (0.0153)	0.7056 (0.0065)	0.6991 (0.0072)	0.7105 (0.0091)	0.7041 (0.0044)
Average	SVM	0.9931 (0.0048)	0.9935 (0.0044)	0.9953 (0.0040)	0.9958 (0.0040)	0.9975 (0.0006)	0.9966 (0.0009)
	MLP	**0.9982** (0.0020)	**0.9982** (0.0013)	**0.9981** (0.0010)	**0.9988** (0.0008)	**0.9994** (0.0005)	**0.9985** (0.0010)
	NB	0.7028 (0.0081)	0.7021 (0.0038)	0.7018 (0.0038)	0.7010 (0.0015)	0.7127 (0.0030)	0.7041 (0.0015)

Table 4. The average recall values of the three methods with different sizes of testing data (e.g., 200, 400, 600, 800 and 10000 from each class) and different size of sample data. The values in the brackets are the standard errors.

Sample	Method	200	400	600	800	1000	Average
10000	SVM	0.8940 (0.0167)	0.8820 (0.0151)	0.8657 (0.0182)	0.8356 (0.0146)	0.6605 (0.0433)	0.8276 (0.0959)
	MLP	0.8885 (0.0216)	0.8855 (0.0140)	0.8718 (0.0083)	0.8501 (0.0198)	0.7949 (0.0194)	0.8582 (0.0385)
	NB	**0.9975** (0.0040)	**0.9945** (0.0047)	**0.9912** (0.0071)	**0.9908** (0.0086)	**0.9803** (0.0077)	**0.9908** (0.0065)
20000	SVM	0.8570 (0.0144)	0.8345 (0.0298)	0.8287 (0.0160)	0.7265 (0.0185)	0.5245 (0.0319)	0.7542 (0.1379)
	MLP	0.9500 (0.0171)	0.9510 (0.0190)	0.9212 (0.0146)	0.8841 (0.0318)	0.8402 (0.0316)	0.9093 (0.0434)
	NB	**0.9950** (0.0055)	**0.9885** (0.0082)	**0.9922** (0.0072)	**0.9874** (0.0082)	**0.9861** (0.0067)	**0.9898** (0.0037)
30000	SVM	0.8695 (0.0151)	0.8458 (0.0169)	0.8020 (0.0279)	0.7375 (0.0415)	0.4577 (0.0382)	0.7425 (0.1669)
	MLP	0.8625 (0.0257)	0.8445 (0.0186)	0.8350 (0.0152)	0.7995 (0.0190)	0.7150 (0.0287)	0.8113 (0.0585)
	NB	**0.9930** (0.0064)	**0.9910** (0.0081)	**0.9940** (0.0066)	**0.9868** (0.0049)	**0.9821** (0.0147)	**0.9894** (0.0049)
40000	SVM	0.8490 (0.0240)	0.8398 (0.0158)	0.7928 (0.0351)	0.6846 (0.0480)	0.4430 (0.0445)	0.7218 (0.1690)
	MLP	0.8445 (0.0224)	0.8370 (0.0172)	0.8182 (0.0187)	0.7826 (0.0221)	0.7014 (0.0242)	0.7967 (0.0584)
	NB	**0.9950** (0.0045)	**0.9860** (0.0051)	**0.9877** (0.0089)	**0.9843** (0.0102)	**0.9866** (0.0087)	**0.9879** (0.0042)
50000	SVM	0.8500 (0.0225)	0.8373 (0.0180)	0.7940 (0.0250)	0.6731 (0.0443)	0.4525 (0.0384)	0.7214 (0.1657)
	MLP	0.8350 (0.0191)	0.8253 (0.0222)	0.8070 (0.0179)	0.7670 (0.0372)	0.7044 (0.0277)	0.7877 (0.0534)
	NB	**0.9950** (0.0074)	**0.9933** (0.0051)	**0.9897** (0.0082)	**0.9881** (0.0090)	**0.9860** (0.0080)	**0.9904** (0.0037)
Average	SVM	0.8639 (0.0187)	0.8479 (0.0195)	0.8166 (0.0310)	0.7315 (0.0642)	0.5076 (0.0913)	0.7535 (0.0437)
	MLP	0.8761 (0.0461)	0.8707 (0.0499)	0.8506 (0.0464)	0.8167 (0.0490)	0.7512 (0.0629)	0.8326 (0.0507)
	NB	**0.9951** (0.0016)	**0.9907** (0.0035)	**0.9910** (0.0024)	**0.9875** (0.0023)	**0.9842** (0.0028	**0.9897** (0.0011)

Table 5. The average F-score values of the three methods with different sizes of testing data (e.g., 200, 400, 600, 800 and 10000 from each class) and different size of sample data. The values in the brackets are the standard errors.

Sample	Method	200	400	600	800	1000	Average
10000	SVM	0.9375 (0.0095)	0.9312 (0.0079)	0.9230 (0.0098)	0.9056 (0.0087)	0.7937 (0.0315)	0.8982 (0.0596)
	MLP	**0.9386** (0.0113)	**0.9378** (0.0079)	**0.9302** (0.0051)	**0.9178** (0.0118)	**0.8851** (0.0120)	**0.9219** (0.0222)
	NB	0.8324 (0.0093)	0.8195 (0.0072)	0.8237 (0.0032)	0.8206 (0.0060)	0.8274 (0.0052)	0.8247 (0.0053)
20000	SVM	0.9217 (0.0074)	0.9073 (0.0180)	0.9039 (0.0086)	0.8408 (0.0126)	0.6868 (0.0271)	0.8521 (0.0975)
	MLP	**0.9269** (0.0099)	**0.9205** (0.0110)	**0.9162** (0.0083)	**0.8982** (0.0192)	**0.8616** (0.0205)	**0.9047** (0.0264)
	NB	0.8246 (0.0090)	0.8205 (0.0095)	0.8194 (0.0071)	0.8196 (0.0068)	0.8286 (0.0030)	0.8225 (0.0040)
30000	SVM	**0.9262** (0.0082)	0.9132 (0.0090)	0.8885 (0.0172)	0.8472 (0.0270)	0.6266 (0.0354)	0.8404 (0.1232)
	MLP	0.9250 (0.0148)	**0.9150** (0.0108)	**0.9092** (0.0093)	**0.8879** (0.0118)	**0.8332** (0.0198)	**0.8941** (0.0366)
	NB	0.8199 (0.0104)	0.8250 (0.0064)	0.8197 (0.0049)	0.8204 (0.0056)	0.8266 (0.0096)	0.8223 (0.0032)
40000	SVM	**0.9164** (0.0147)	**0.9113** (0.0092)	0.8836 (0.0217)	0.8110 (0.0336)	0.6122 (0.0422)	0.8269 (0.1272)
	MLP	0.9150 (0.0136)	0.9108 (0.0102)	**0.8997** (0.0112)	**0.8777** (0.0141)	**0.8242** (0.0167)	**0.8855** (0.0372)
	NB	0.8167 (0.0110)	0.8219 (0.0071)	0.8214 (0.0089)	0.8200 (0.0065)	0.8246 (0.0088)	0.8209 (0.0029)
50000	SVM	**0.9170** (0.0124)	**0.9102** (0.0110)	0.8841 (0.0152)	0.8030 (0.0311)	0.6218 (0.0359)	0.8272 (0.1234)
	MLP	0.9097 (0.0110)	0.9040 (0.0133)	**0.8925** (0.0113)	**0.8675** (0.0239)	**0.8261** (0.0191)	**0.8799** (0.0342)
	NB	0.8246 (0.0090)	0.8218 (0.0104)	0.8238 (0.0058)	0.8188 (0.0038)	0.8258 (0.0064)	0.8230 (0.0028)
Average	SVM	**0.9238** (0.0086)	0.9146 (0.0095)	0.8966 (0.0169)	0.8415 (0.0405)	0.6682 (0.0760)	0.8490 (0.0294)
	MLP	0.9230 (0.0112)	**0.9176** (0.0128)	**0.9096** (0.0147)	**0.8898** (0.0194)	**0.8460** (0.0265)	**0.8972** (0.0167)
	NB	0.8236 (0.0059)	0.8217 (0.0021)	0.8215 (0.0022)	0.8197 (0.0007)	0.8270 (0.0018)	0.8227 (0.0014)

simulation results based on public PaySim dataset show that the three methods perform satisfactorily. In term of effectiveness, MLP should be the first choice among the three, while NB is the most efficient method among them. Moreover, NB does not have hyper-parameters to tune, while in SVM and MLP we need to tune some parameters to obtain the optimal results.

Data are still the main challenges in the research of this topic. By comparing different methods on only one data is not a sound comparison. We hope that in the future there are more available dataset to be tested. Moreover, the three methods discussed in this paper are quite traditional, however, they perform well. With new data, they may not be that effective, then more advanced methods are needed to be developed. We leave this problem to the future work. Fraud detection problem attract more interests of commercial industry than academia. One of the reason is the shortage of public data for systematic and scientific comparison, while commercial industry are the ones that hold the data. Making data available for research may attract more interests from the researchers in community.

References

1. Bolton, Richard J and Hand, David J. Statistical fraud detection: A review. Statistical science. 235–249, (2002).
2. Chan, Philip K and Fan, Wei and Prodromidis, Andreas L and Stolfo, Salvatore J. Distributed data mining in credit card fraud detection. IEEE Intelligent Systems and Their Applications vol (14) 6, 67–74 (1999).
3. Phua, Clifton and Lee, Vincent and Smith, Kate and Gayler, Ross. A comprehensive survey of data mining-based fraud detection research. arXiv preprint arXiv:1009.6119. (2010).
4. Sudjianto, Agus and Nair, Sheela and Yuan, Ming and Zhang, Aijun and Kern, Daniel and Cela-Díaz, Fernando. Statistical methods for fighting financial crimes. Technometrics. vol (52) 1, 5–19 (2010).
5. Lopez-Rojas, Edgar and Elmir, Ahmad and Axelsson, Stefan. PaySim: A financial mobile money simulator for fraud detection. 28th European Modeling and Simulation Symposium, EMSS, Larnaca. 249–255 (2016).
6. Ngai, Eric WT and Hu, Yong and Wong, YH and Chen, Yijun and Sun, Xin. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. Decision Support Systems, vol(50) 3, 559–569 (2011).
7. Haykin, Simon S and Haykin, Simon S and Haykin, Simon S and Haykin, Simon S. Neural networks and learning machines. vol (3) (2009) Pearson Upper Saddle River, NJ, USA.
8. Dorronsoro, Jose R and Ginel, Francisco and Sánchez, Carmen R and Santa Cruz, Carlos. Neural fraud detection in credit card operations. IEEE transactions on neural networks (1997).
9. Fanning, Kurt M and Cogger, Kenneth O. Neural network detection of management fraud using published financial data. Intelligent Systems in Accounting, Finance & Management, vol(7) 1, 21–41 (1998).
10. Viaene, Stijn and Dedene, Guido and Derrig, Richard A. Auto claim fraud detection using Bayesian learning neural networks. Expert Systems with Applications, vol(29) 3, 653–666, (2005).

11. Bermúdez, Ll and Pérez, JM and Ayuso, M and Gómez, E and Vázquez, FJ. A Bayesian dichotomous model with asymmetric link for fraud in insurance. Insurance: Mathematics and Economics. vol(42) 2, 779–786, (2008).
12. Viaene, Stijn and Derrig, Richard A and Dedene, Guido. A case study of applying boosting Naive Bayes to claim fraud diagnosis. IEEE Transactions on Knowledge and Data Engineering. vol(16) 5, 612–620, (2004).
13. Subudhi, Sharmila and Panigrahi, Suvasini. Quarter-sphere support vector machine for fraud detection in mobile telecommunication networks. Procedia Computer Science. vol (48) , 353–359, (2015).
14. Şahin, Yusuf G and Duman, Ekrem. Detecting credit card fraud by decision trees and support vector machines. Proceedings of the International MultiConference of Engineers and Computer Scientists , 442-447, (2011).
15. Dheepa, V and Dhanapal, R. Behavior based credit card fraud detection using support vector machines. ICTACT Journal on Soft computing. vol(4) 4, 391–7, (2012).
16. Choi, Dahee and Lee, Kyungho. Machine Learning based Approach to Financial Fraud Detection Process in Mobile Payment System. IT CoNvergence PRActices (INPRA), vol. (5) 4, 12-24, (2017).
17. Kou, Yufeng and Lu, Chang-Tien and Sirwongwattana, Sirirat and Huang, Yo-Ping. Survey of fraud detection techniques. Networking, sensing and control, 2004 IEEE international conference on. vol (2) 749–754, (2004).
18. Yue, Dianmin and Wu, Xiaodan and Wang, Yunfeng and Li, Yue and Chu, Chao-Hsien. A review of data mining-based financial fraud detection research. Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on, 5519–5522, (2007).
19. Richhariya, Pankaj and Singh, Prashant K. A survey on financial fraud detection methodologies. International journal of computer applications. vol. (45) 22, 15–22, (2012).
20. Kendall, Maurice George. The advanced theory of statistics. The advanced theory of statistics., 2nd Ed Charles Griffin and Co., Ltd., London, (1946).
21. Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, vol. (12), 2825–2830, (2011).
22. Manevitz, Larry M and Yousef, Malik. One-class SVMs for document classification. Journal of machine Learning research, vol. (2) Dec, 139–154, (2001).
23. Aizerman, Mark A. Theoretical foundations of the potential function method in pattern recognition learning. Automation and remote control, vol. (25), 821–837, (1964).
24. Vert, Jean-Philippe and Tsuda, Koji and Schölkopf, Bernhard. A primer on kernel methods. Kernel methods in computational biology, vol. (47), 35–70, (2004).
25. Cortes, Corinna and Vapnik, Vladimir. Support-vector networks. Machine learning. vol. (20) 3, 273–297, (1995).
26. Hahnloser, Richard HR and Sarpeshkar, Rahul and Mahowald, Misha A and Douglas, Rodney J and Seung, H Sebastian. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature, vol. (405) 6789, 947, (2000).
27. Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10) 807–814, (2010).