Thinking Like A Fraudster: Detecting Fraudulent Transactions Via Statistical Sequential Features*

Chen Jing[1,2,3], Cheng Wang[1,2,3], and Chungang Yan[1,2,3]

[1] Department of Computer Science and Technology, Tongji University, Shanghai, China

[2] Key Laboratory of Embedded System and Service Computing, Ministry of Education, Shanghai, China

[3] Shanghai Electronic Transactions and Information Service Collaborative Innovation Center

{jingchen23, cwang}@tongji.edu.cn, cgyan2@163.com

Abstract. Aiming at the increasing threat of fraud in electronic transactions, so far researchers have already proposed many different models. However, few previous studies take advantage of the sequential characteristics of fraudulent transactions. In this paper, by statistical analysis on a real dataset, we discover that partial-order sequential features are able to reflect the intrinsic motivation of fraudsters, e.g., stealing the money as quickly as possible before being intercepted. Based on the sequential features, we propose a novel model, *SeqFD (Sequential feature boosting Fraud Detector)*, to detect fraudulent transactions real-timely. *SeqFD* applies a sliding time window strategy to aggregate the historical transactions. In specific, statistical sequential features are computed based on the transactions within the time window. Thus, the raw dataset can be transformed into a feature set. Several classification models are evaluated on the feature set, and finally, XGBoost is validated to be a fast, accurate and robust classifier which fits well with *SeqFD*. The experiments on real dataset show that the proposed model reaches a 97.2% *TPR (True Positive Rate)* when *FPR (False Positive Rate)* is less than 1%. Furthermore, the average time for giving a prediction is 1.5 milliseconds, which meets the real-time requirement in the industry.

Keywords: Fraudulent Transaction Detection · Statistical Sequential Features · Sliding Time Window · Machine Learning.

1 Introduction

1.1 Background

Recent years, the huge development in e-commerce makes more and more people shop online. The amount of money people spend online is also increasing. In

* This work was supported in part by the National Natural Science Foundation of China under Grant 61332008, in part by the National Key Research and Development Program of China under Grant 2018YFC0831403.

China's 2017 Double Eleven shopping festival, the total sales on Tmall[4] reached 168.2 billion yuan [1]. However, this prosperity gives criminals chances to steal money. According to the Nilson Report in October 2016, worldwide losses from credit card fraud rose to 21 billion dollars in 2015, and will possibly reach 31 billion dollars by 2020 [13]. There is no doubt that it is meaningful to prevent people's properties from being stolen by fraudsters.

Although the criminals have various tricks, such as telephone fraud, Trojan and pseudo base station [7], their unchanged goal is to steal money from people's bank accounts. It is spontaneous to think about how to prevent fraud when fraud is presenting as fraudulent transactions. In the literature, fraud detection methodologies using behavioral models are proposed [5, 2]. Behavioral models mainly fall into two categories: individual behavioral models and crowd behavioral models. For the individual models, concept drift [17] and lack-of-history are the two main tough challenges faced by individual behavioral models. Basically, concept drift means that a change in behavior may not be due to fraud[10]. The lack-of-history problem is that a portion of customers do not have sufficient historical records to depict their behavioral patterns. To handle concept drift, one common solution is using the time window strategy [11] to neglect the old transaction and keep the model updated. This strategy will make the lack of history problem severer, but if we set a larger time window to contain more records, it contradicts the original intention to solve the problem of concept drift.

As a result, researchers are trying to build models based on crowd behavior. In general, their models extract normal and fraud behavioral patterns from a large number of customers. By measuring the differences between the two patterns, effective features can be designed, then a classifier or an anomaly detector can be trained. Crowd behavioral models alleviate the lack of history problem because the history of active customers can be used to make up for that of inactive customers. In the literature, many crowd behavioral models have been proposed for e-transaction fraud detection. However, most of the previous studies have one or more of the three problems below.

Lack of sequential features. In order to build a behavioral model, sequences of transactions are supposed to be taken into consideration rather than isolated transactions. In order to avoid dimension disaster, transaction aggregation strategies are applied to build the behavioral models [18]. A common way for transaction aggregation is to set up several time spans, and calculate the statistical values of spending for each time span, such as average and variance, as aggregated features [6, 7]. However, in previous studies, the aggregated features do not contain information about the partial-order relationship between consecutive transactions. Thus they are not sequential features. By real data analysis, we find out that the sequential features are strongly correlated with fraud. In Section 2 this will be explained in detail.

Lack of efficiency tests. Recent studies also tried to implement more complex models, such as deep learning models, to detect fraudulent transactions. For example, CNN (Convolutional Neural Network) is used for credit card fraud

[4] Tmall is a Chinese-language website for business-to-consumer (B2C) online retail

detection in recent studies [12, 6]. However, most of them ignored the efficiency test. Deep learning models are probably not fast enough to meet the requirement of efficiency, because nowadays a deep learning model usually contains at least hundreds of thousands of units [8], needs GPUs and parallelized clusters to compute. As a result, both the training and predicting process could take a long time. On the contrary, banks usually require their fraud detection systems to give a response in a couple of milliseconds, and the training time should also be as short as possible.

Lack of real dataset. Banks are usually very sensitive to the confidentiality of their data, this makes the publication of a real electronic transaction dataset nearly impossible [16]. Therefore, some previous studies used simulated datasets to train and test their models. Nevertheless, the results obtained from simulated datasets might be inconvincible. In addition, some of the previous studies which used simulated dataset employed *Accuracy* as a performance indicator [4]. However, *Accuracy* is an unsuitable indicator in the area of fraud detection, because real datasets are highly unbalanced [15]. This paper applies *TPR (True Positive Rate)* and *FPR (False Positive Rate)* as the indicators.

1.2 Our Work

In this paper, by statistical analysis, we discover that sequential features are a reflection of the intrinsic motivation of the fraudsters. We apply a sliding time window to aggregate the simple sequential features into statistical sequential features. Based on These features, we propose a real-time fraudulent transaction detection model, *SeqFD (Sequential feature boosting Fraud Detector)*. By experiments on a real dataset, we validate that *SeqFD* can detect more than 97% fraudulent transactions but only disturb less than 1% normal transactions, and it can give a prediction in 1.5 milliseconds on average. To choose a suitable classifier, we conduct experiments on six machine learning models. Finally, XGBoost is validated to be a suitable classifier for *SeqFD*. Moreover, experiments are also conducted to choose a feasible window size, and to test the influence of under-sampling.

At the same time of presenting *SeqFD* in detail, a practical workflow for fraudulent transaction detection is presented. After *SeqFD* gives a prediction, staffs of banks can check the suspicious transactions by phone call, so the confirmed fraudulent transactions can be labeled. With continuously new-coming labeled instances, *SeqFD* can be trained periodically to be kept updating. Our contributions are summarized as follows:

- We design novel statistical sequential features which are effective for fraudulent transaction detection, and which can reflect the intrinsic motivation of fraudsters.
- We propose and implement *SeqFD*. By comprehensive experimental evaluation, we prove the effectiveness and the usability of *SeqFD*.
- Based on real data, we present our statistical discoveries and learned lessons of fraudulent transactions, which are valuable for future studies.

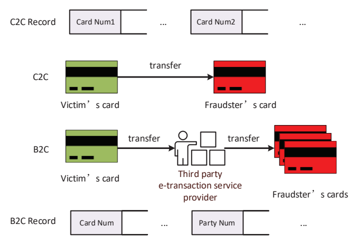


Fig. 1. The difference between fraudulent transactions in B2C and C2C scenarios.

The rest of this paper is organized as follows: Section 2 offers the statistical analysis and the learned lessons of the dataset. In Section 3, the mechanism of *SeqFD* is demonstrated. In Section 4, the results of the experiments are shown. Section 5 gives a complimentary discussion on *SeqFD* and Section 6 draws a conclusion for this paper.

2 Real Data Analysis

In this section, the dataset we study is introduced in detail first. Afterward, the statistical characteristics of the fraudulent transactions are shown graphically. Based on the abnormal patterns we observe from the fraudulent transactions, we explore the possible motivations behind their fraudulent behaviors.

2.1 Dataset Description

We study a real electronic transaction dataset provided by a real commercial bank. The dataset totally contains 3502048 B2C transaction records made by 92133 customers in 3 months (from April 1, 2017, to June 30, 2017). Among all the transaction records, 65291 are labeled fraudulent manually by staffs of the bank. Although this dataset only covers a small portion of all the customers of the bank, it covers all the customers who were defrauded in the 3-month time span. Among the 92133 customers, 8238 are victims, and all of their transactions in the time span are extracted into our dataset. The other normal customers are picked out randomly, and all of their transaction records in the time span are extracted. These two parties form the whole dataset.

Although the original dataset has more than 20 attributes, some of the attribute values are missing in many instances, and some of the attributes have the same value for all the instances. We omit those helpless attributes. After data preprocessing, for each transaction record, we preserve 8 attributes. Their explanations are listed in Table 1. Note that Customer ID and Vendor ID do not have the same format.

2.2 Learned Lessons

Why B2C other than C2C Besides the B2C transaction records, actually we are also provided the C2C transaction records of those 92133 customers in the

Table 1. Attributes explanation

Attribute Name	Type	Representation
Customer ID	String	A unique customer.
Transaction ID	String	A unique transaction record.
Vendor ID	String	A shop, a restaurant or a third-party payment provider, etc.
Transaction Time	Date	The exact time when the transaction occurred.
Daily Limit	Numeric	The daily spending upper limit of an account.
Single Limit	Numeric	The spending upper limit for one transaction.
Transaction Amount	Numeric	The amount of money that the customer pay.
Frequently-Used IP Address	Boolean	If the transaction comes from an IP address which is frequently used by the customer.

three months, but the number of fraudulent instances in those C2C transactions is only 1. Why criminals steal money mainly by B2C transactions instead of C2C transactions? Regarding the difference of their procedures shown in Figure 1, the explanation is as follows: If a fraudster transfers the money into his/her own card directly by a C2C transaction, it will be too risky because his/her card number can be seen in the C2C transaction record. But in a B2C scenario, B can be a non-bank e-transaction service provider, such as PayPal. For example, a fraudster can transfer money from the stolen card to a PayPal account for the first step and then transfer the money from the PayPal account to multiple fraudulent cards. In the B2C scenario, the PayPal account and the fraudulent cards are untraceable in the transaction record, because in a B2C transaction record, the Vendor Id field only contains a String that stands for the PayPal company, not the specific PayPal account. By stealing money via B2C transactions, a fraudster can keep himself/herself invisible in the transaction records. This is probably the reason why fraudsters usually commit crimes through B2C other than C2C transactions.

Statistical discoveries We conduct statistical analysis on a month of transactions. We find out that 99.6% of the fraudulent transactions did not use a frequently-used IP address (FUIP), as shown in Figure 2. This means that these transactions are highly possible to be made by the criminals after they have stolen the accounts, instead of being made by the deceived victims themselves. We also discover that more than 96% of the fraudulent B2C transactions are not the first transaction made by the customer in the month (without the influence of data boundary, this ratio could be higher). Furthermore, we discover that the time intervals between each two consecutive fraudulent transactions of a customer are often abnormally short, and the amounts of the two transactions are often close. We name these two quantities *TTD* (Transaction Time Difference) and *TAD* (Transaction Amount Difference), so the graph of the joint cumulative density function is drawn in Figure 3. It is obvious that fraudulent transactions usually have smaller *TTD* and *TAD* than normal transactions. In another word, fraudsters usually steal the money with multiple consecutive quick transfers. Another abnormal phenomenon about the fraudulent transactions we find out is shown in Figure 4. The amounts of normal transactions present a power-law

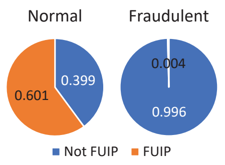


Fig. 2. Distribution of the transaction records when they are divided by the frequently-use IP address attribute.

distribution, by contrast, the amounts of fraudulent transactions present a quite different distribution which has several isolated peaks. The two dominating peaks are 1000 and 2000.

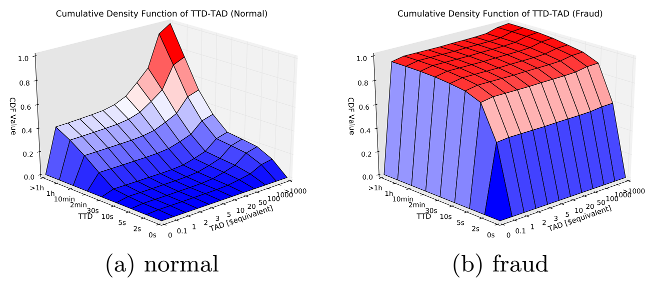


Fig. 3. The TTD-TAD distribution.

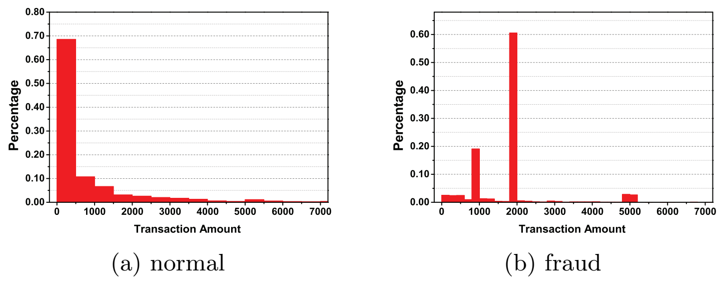


Fig. 4. The difference between fraudulent and normal transactions in amount distribution.

Explore fraudsters' motivation So far we have presented the observations and discoveries we obtained from the statistical results. In this part, we want to explore the reasons behind the abnormal fraudulent behavioral patterns.

Let's start with the abnormally small *TTD*. Consider such a scenario: When a fraudster has just stolen an account, his goal is to take more money away.

Table 2. Frequencies of different daily spending upper limits.

Daily Spending Upper Limit	Frequency
1000	1.65%
2000	10.80%
5000	80.60%
10000	1.40%
20000	5.54%
50000 or more	0.07%

However, in most instances, as soon as a transfer occurs, the customer will be informed by the bank immediately probably by phone message. Then the customer will immediately inform the bank to freeze the account. This leads to the abnormal small *TTD*, because if the fraudsters want to steal more money, they have to do it quickly. They don't even have time to camouflage the TTD. But why the fraudsters do not steal money in a single transaction? Why the fraudulent transactions often present in sequences?

Usually, a bank will set a daily spending upper limit for each customer. We make statistics for all the upper limits on a month of transaction records, which is shown in Table 2. It shows that over 80% of the customers have a daily upper limit of 5000. This fact can explain the two peaks (1000 and 2000) in Figure 4(b): Consider a criminal has stolen a bank account and does not know whether or not the card has been used for shopping on the very day. In such a situation, 1000+2000+2000 could be one good combination of transfer amounts because of three advantages: First, compared to a single transfer with larger amount, such as 5000, 1000+2000+2000 is more likely to succeed, at least partially, because more than 93% of the customers have a daily spending limit equal to or less than 5000. Once the customers have used their card for shopping today, a transfer request of 5000 will fail, because it exceeds the daily spending upper limit. Second, once the card is not used yet on the very day, then this combination can take as much money as possible without any remnant. Third, compared to transfer amount less than 1000, this combination needs fewer transaction requests, which leads to fewer risk of being intercepted by fraud detection systems.

To summarize, fraudsters are willing to steal the money in smaller amounts by multiple quick transfers, because this way has a higher profit expectation and fewer risks. As a result, the fraudulent transactions present a sequential form for most of the time, and the time intervals between every two consecutive transactions are often small.

3 Proposed Model

In response to the threat of fraudulent B2C transactions, we propose a novel model, *SeqFD*, for real-time fraudulent transaction detection. The innovation of *SeqFD* lies in the statistical sequential features. In this section, we first overview the mechanism and deployment scenario of *SeqFD*. Then, we elaborate on how to compute the statistical sequential features by using the sliding time window. Finally, we list all the 9 features applied by *SeqFD*.

3.1 Overview

The workflow of *SeqFD* is depicted in Figure 5. Within the dashed line in the right part, the mechanism of *SeqFD* is demonstrated in detail. In the left part, a possible deployment scenario is shown.

The mechanism of SeqFD *SeqFD* has two stages: the training stage and the classification stage. At the training stage, the labeled transaction records will be sampled to form a raw dataset. The reasons for sampling are two-fold: First, the volume of the whole dataset is very large, using the whole dataset to train the classifier is quite time-consuming. Second, in reality, fraudulent instances are often far less than normal instances so the dataset is extremely skewed. One feasible sampling strategy is just like how the dataset is extracted for our study: Obtain all the victims in a time span first (we use three months in this work), then query for all the transactions these customers have made within the time span. After that, randomly pick out some normal customers and retrieve their transaction records in the same time span. Finally, combine these two parts to form a labeled training set, which is not that large or skewed.

Through the sliding time window strategy for transaction aggregation, a raw instance can be turned into a feature vector which includes sequential features. Feed the feature set to a machine learning model for training, and after that, a trained classifier will be ready to give a prediction. At the classification stage, streaming transaction requests will be sent to *SeqFD*, and *SeqFD* will give a real-time response. In specific, *SeqFD* will first turn the raw request into a feature vector composed of the same features as above, and then the feature vector will be sent to the classifier. Finally, the classifier gives the prediction.

Deployment scenario of *SeqFD* The left part of Figure 5 shows how to put *SeqFD* into a real application. When *SeqFD* receives a transaction request, it gives a prediction. If *SeqFD* judges a transaction request suspicious, people of the bank could give a phone call to the customer to figure out if the transaction is truly fraudulent. Then the transaction can be labeled and be added into the database of labeled transaction records. Another source of labeled instances is police reports. *SeqFD* cannot catch all the fraudulent transactions, and some of the missing fraudulent instances might be obtained from the police. With these two sources of labeled transactions, *SeqFD* can be retrained periodically to make the model updated. Therefore, *SeqFD* is able to adjust itself in accordance with the change in crowd behavior patterns. No matter how the normal crowd behavior change, as long as it is different from the fraudulent crowd behavior, then *SeqFD* is able to work effectively.

3.2 Sliding Time Window

This subsection introduces the detailed design of the transaction aggregation technique based on sliding time window strategy. In the database of historical

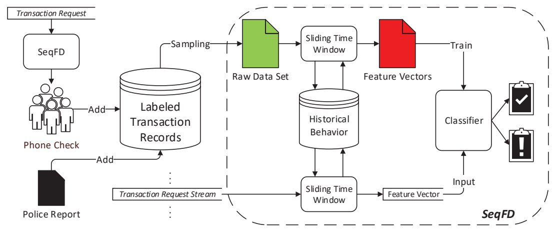


Fig. 5. *SeqFD* overview.

behavior which is in the middle of Figure 5, an independent list of historical transactions is kept individually for every customer. An example of such a list of a certain customer is shown in Figure 6. In this simple example, the size of the time window is set to 1 minute, which means that the list will only contain the newest transaction records that happened within 1 minute ago. Every time a new-coming transaction is added into the list, the time window slides forward, then the obsolete transactions will be thrown out of the time window. Sliding time window ensures that the features computed by aggregated transactions can precisely depict the recent behavior pattern of a customer.

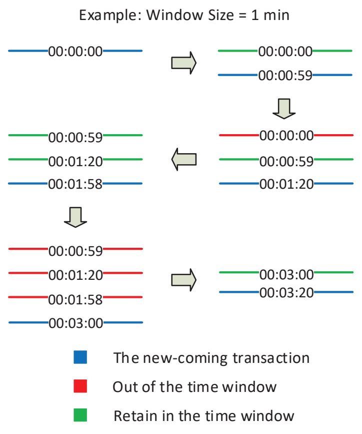


Fig. 6. An exemplary illustration of how the time window slides on the historical transactions of a certain customer.

3.3 Feature Engineering

The features and their explanations are shown in Table 3.

As mentioned in Section 2.2, *TTD* and *TAD* are two typical sequential features, but they only include information on two consecutive transactions. What

Table 3. Selected Features

Name	Type	Explanation	Sequential?	Original?
Amount	Numeric	The amount of the transaction.	No	Yes
FUIP	Boolean	If this transaction is made from a frequently used IP address.	No	Yes
OverLim	Boolean	If this transaction is over the spending limitation.	No	No
AmtAvg	Numeric	The average of the transaction amounts in the time window.	No	No
Times	Numeric	The number of transactions within the time window.	No	No
TDAvg	Numeric	The average of all the TTDs within the time window.	Yes	No
TDVar	Numeric	The variance of all the TTDs within the time window.	Yes	No
ADAvg	Numeric	The average of all the TADs within the time window.	Yes	No
ADVar	Numeric	The variance of all the TADs within the time window.	Yes	No

we need are features that can summarize a bunch of transactions, so the sequential features should be aggregated. In specific, within the time window, the *TTD*s and *TAD*s are aggregated by computing their average and variance. Thus, statistical sequential features which summarize the characteristics of all the transactions in the time window are generated.

In this work, we design only 9 features, and most of them can be computed just by time and amount. This makes our model easy to be transplanted to other datasets. Furthermore, *SeqFD* can also sufficiently protect the privacy of customers. If a bank deploys *SeqFD* to help them detect fraudulent transactions, *SeqFD* will only gather the basic information of the transactions. No personal information will be gathered. This is also one of the advantages of *SeqFD*.

4 Evaluations

In this section, comprehensive experiments are conducted to evaluate *SeqFD*. The questions for which we want to find out answers are as follows:

(1) Which machine learning model performs best on the features?

(2) What size is appropriate for the sliding time window?

(3) Are the statistical sequential features really important?

(4) Are there any possible negative influences when the under-sampling is applied to solve the skewed-data problem?

4.1 Experimental Setup

We use the dataset of April and May as the training set and the dataset of June as the test set. *Cross-validation* is not adopted in the experiment because it will cause the *time travel problem*, e.g., using the data from future to train a model and using the data from past to test. The training set contains 2393817 normal instances and 40393 fraudulent instances, and the testing set contains 1003539 normal instances and 24898 fraudulent instances. The ratio of the two classes seem approximately 59 : 1, but it is not the true ratio because 2393817 is just a portion of the normal instances. Actually, the number of transactions in a month is nearly 14 million, so the actual ratio is nearly 693 : 1. As the two classes are highly unbalanced, we sample 10% of the normal instances.

Note that we do the sampling process after the raw dataset have been transformed to feature set so that the statistical sequential features are kept lossless. The instances of the test set are sent to the classifier in the right temporal order to simulate the transaction stream.

Window size candidates We prepare six candidate window sizes on different scales: 1 minute, 10 minutes, 1 hour, 1 day, 1 week and 1 month. Our goal is to figure out which one can lead to the best performance of classification. The window size will be referred to as *WS* for short afterward. A *WS* larger than 1 month is not taken into account since the intervals of the adopted dataset for training is only two months.

Machine learning model candidates For classification, the candidate machine learning models we choose are the Random Forest [9], XGBoost [3], Decision Tree, Naive Bayes, 2-hidden-layer Neural Network, and Logistic Regression. For Neural Network, we set 10 perceptrons to each hidden-layer. For other models, we use the default hyper-parameters preset in Scikit-Learn [14].

Assessment criteria We use three assessment criteria to evaluate the *WS* and choose a suitable classifier.

AUC (Area Under ROC Curve): For each window size, we take the *AUC* of the machine learning models as a performance indicator.

The highest *TPR* when *FPR* is less than 0.01: According to our dataset provider, in the industry 1% is the tolerance upper limit for *FPR*. As a result, a *TPR* reached with a *FPR* higher than 1% is regarded as meaningless in this work.

The time for generating features: For a practical model, the efficiency, e.g., the time of generating features, is a necessary factor.

4.2 Evaluation Results

Figure 7 shows that XGBoost has the highest AUC for all the six candidate window sizes, it performs robustly and stably. Random Forest is able to take the second place except when *WS* is 1 minute. The performances of the other four machine learning models fall behind.

Figure 8 presents the highest *TPR* reached by each machine learning model when *FPR* is controlled under 0.01 by tuning the classification threshold. The best record is 97.2%, which is achieved by XGBoost when *WS* is set to be 1 month. Random Forest performs better than XGBoost when the *WS* is smaller, except for 1 minute. And the other four models still fall behind.

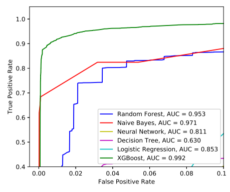
In common sense, a larger window size could lead to more cost for efficiency, because for the customers who make transactions frequently and continuously, a larger window size will contain more transactions to compute. Thus we test the average time it costs to transform a raw transaction into a feature vector on a server with a dual-core 2.40 GHz CPU and 32 GB RAM, and the result is shown in Figure 9. Surprisingly, it shows that when the *WS* is 1 month, the average

Table 4. The training and predicting time (seconds) of the candidate ML models.

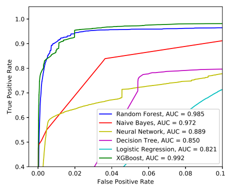
Model	Training Time	Predicting Time
Random Forest	7.18	1.79
Naive Bayes	0.94	0.83
Decision Tree	6.38	0.62
Logistic Regression	7.05	0.67
XGBoost	9.44	1.09
Neural Network	32.96	0.78

time is only about 1.5 milliseconds, which is not far more than the average time when the *WS* is smaller. Therefore, 1 month is adopted as the appropriate *WS*.

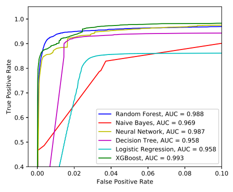
Furthermore, we also test the training time and the predicting time of each machine learning model. These two indicators are not influenced by the *WS*, because the number of instances and the number of features remain unchanged when *WS* is changed. Therefore, we set the *WS* to be 1 month and test the training and test efficiency for each classifier. The result is shown in Table 4. For the training stage, the time cost of XGBoost is 9.44 seconds. For the test stage, the test set contains 1028437 instances, so the average predicting time of XGBoost for one instance is less than 2 microseconds. Therefore, the total time cost for the predicting stage is $1.5ms + 2\mu s \approx 1.5ms$. According to our experience of the industry, this time cost is far below the tolerance upper bound. XGBoost is nearly 40% faster than Random Forest. In addition, XGBoost has the best performance when the *WS* is 1 month. Therefore, it should be applied by *SeqFD*.



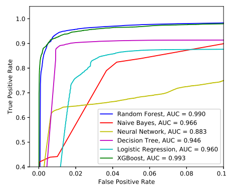
(a) 1 min



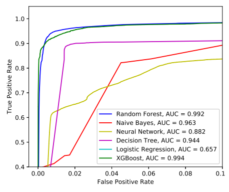
(b) 10 min



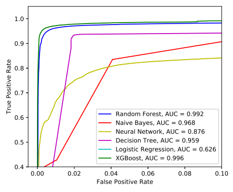
(c) 1 hour



(d) 1 day



(e) 1 week



(f) 1 month

Fig. 7. The ROC curves of the candidate machine learning models under different window sizes.

4.3 Feature Importance

As XGBoost and Random Forest are both capable to output the importance of the features, we use them to evaluate the importance of the features. The result is shown in Figure 10 (a).

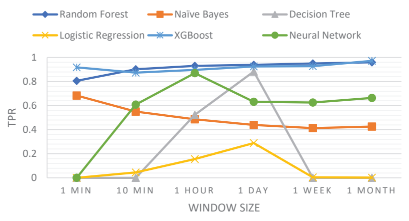


Fig. 8. The highest TPR reached by the candidate machine learning models when FPR is less than 0.01.

Figure 10 (a) shows that all of the 4 statistical sequential features play indispensable roles in *SeqFD*, especially *TDAvg*. The other two features generated by the sliding time window, *Times* and *AmtAvg*, are also quite important. For the original features, *Amount* plays an important role in both models. *FUIP* is quite important in Random Forest but seems useless in XGBoost. We also conduct a comparative experiment with different features. For the first set, all the nine features are included; For the second set, the four statistical sequential features are eliminated; For the third set, transaction aggregation is not used and only two original features, *Amount* and *FUIP*, are included. The result in Figure 10 (b) shows that the statistical sequential features truly boost the performance of *SeqFD*.

4.4 Effects of Under-Sampling

In the evaluations above, we use a training set composed of 10% normal instances (about 100000) and all fraudulent instances (24898). Actually, 10% is out of intuition. Therefore, we want to figure out if different sampling ratios will lead to fluctuations in performance. We use the ROC of XGBoost to represent the classification performance. The *WS* is set to be 1 month. The candidate sampling ratios for the normal instances are 5%, 10%, 20%, 50% and non-sampling, respectively. The result is shown in Figure 11.

Figure 11 shows that all of the candidate sampling ratios have an AUC larger than 0.995, which means that *SeqFD* performs robustly without a large fluctuation under different sampling ratios. Surprisingly, when the sampling ratio is set to be 5%, XGBoost performs the best. Although this result may be due to fortuitous, the rationality of under-sampling for solving the skewed-data problem is proved.

5 Discussion

Although $TPR = 97.2\%$ with $FPR < 1\%$ is the highest performance we get from the experiment, actually the performance can be improved further in practice because our dataset has a boundary. Some of the seemingly first transactions in sequences are actually not the first ones, because those plausible headers could have precedents in the data outside the boundary. For example, when we

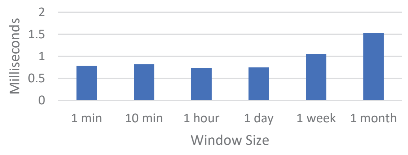
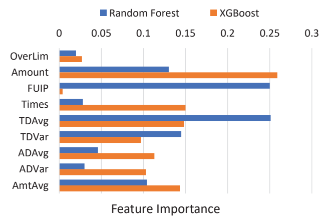


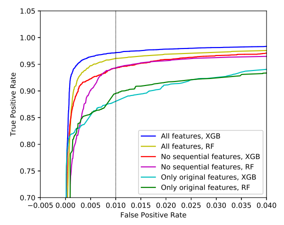
Fig. 9. The average time it costs to transform a raw transaction to a feature vector.

compute the feature vectors for the transactions in April, the transactions of March should be within the time window if *WS* is set to be 1 month. But we are not provided the dataset of March. As a result, a portion of the feature vectors, especially the ones near the boundary, are not computed precisely. In practice, the transactions will be continuous streaming data. The problem of data boundary can be diluted infinitely.

Another issue is about the test set. As mentioned before, the test set contains a sampled portion of the normal transactions. However, as the normal transactions are picked randomly, they are supposed to have the same distribution as the whole transaction dataset. Thus, there is no straightforward reason for the *FPR* to rise if the whole transaction dataset is used to test *SeqFD*, because with the rise of the numerator (the number of error-alarmed transactions), the denominator (the number of all the normal transactions) is also rising. In addition, as our test set contains all the fraudulent transactions in the time span, the *TPR* has absolutely no reason to decrease when using the whole dataset to test *SeqFD*. Thus the performances of *SeqFD* in the evaluations are reliable indeed.



(a) Feature importances.



(b) The performance reached by different features.

Fig. 10. Evaluations of the feature importance. In common sense, when the performance gets higher, it will be harder to get improved.

The third point we want to discuss is the scalability of *SeqFD*. For the classification process of *SeqFD*, the top two classifiers, Random Forest and XGBoost, are both scalable models [9, 3]. For the feature generation process of *SeqFD*, the historical records retained by the sliding time window of two customers have no coupling, so transactions of different customers can also be transformed into feature vectors in parallel. We use Redis to implement the data warehouse of

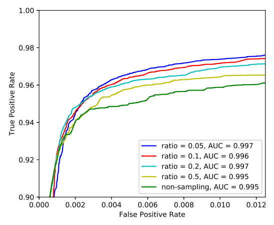


Fig. 11. The ROC curves of XGBoost when the *WS* is 1 month under different sampling ratios of normal instances.

the historical records. The Redis data warehouse can be distributed in multiple servers in cluster mode, and it has high efficiency for both reading and writing. As a result, *SeqFD* is scalable and is possible to be applied to real bank systems with a huge volume of data.

Our work in progress is to deploy *SeqFD* in a distributed structure for real banking transaction systems with high concurrency and burst network traffic, and to test the performance of *SeqFD* in real streaming data.

6 Conclusion

To reach the goal of detecting effectively fraudulent transactions, we propose a novel model, *SeqFD*. The main innovation of *SeqFD* is a new-designed set of statistical sequential features. Instead of intuition, the sequential features come from statistical analysis of a real dataset. By observing the difference between normal and fraudulent transactions, we discover several typical abnormal patterns of the fraudulent transactions which can be distinguished by sequential features. We explore the reasons behind the observations, and find out that the fraudulent behavior patterns are possibly caused by the fraudsters' intrinsic motivation: They always want to steal more money as quickly as possible. Accordingly, we apply a sliding time window strategy to aggregate the sequential features into statistical sequential features, and we present the mechanism and deployment scenario of *SeqFD* in detail.

Through experimental evaluations, among the six representative machine learning models, we find that XGBoost is the classifier which fits the best with *SeqFD*. In specific, when the window size is set to be 1 month, XGBoost reaches an AUC of 0.996, and a TPR of 97.2% when FPR is less than 1%. In *SeqFD*, the problem of concept drift is alleviated by the sliding time window, and the problem of class-imbalanced data is solved by the under-sampling. The experiment shows that no observable negative influence emerges when the sampling ratio is smaller. Furthermore, the result of the efficiency test is also given. On a server with dual-core 2.40 GHz CPU and 32 GB RAM, the trained *SeqFD* can give a response to a transaction request in about 1.5 milliseconds, which can competently meet the requirement of real-time. This makes *SeqFD* a practical model for fraudulent transaction detection in real-world applications.

References

1. China double 11 shopping festival sales statistics 2017 (2017), https://www.chinainternetwatch.com/22791/double-11-2017/
2. Bolton, R.J., Hand, D.J.: Statistical fraud detection: A review. Statistical Science **17**(3), 235–249 (2002)
3. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp. 785–794. ACM (2016)
4. Duman, E., Elikucuk, I.: Solving credit card fraud detection problem by the new metaheuristics migrating birds optimization. In: International Conference on Artificial Neural Networks: Advences in Computational Intelligence. pp. 62–71 (2013)
5. Fawcett, T., Provost, F.: Adaptive fraud detection. Data Mining and Knowledge Discovery **1**(3), 291–316 (1997)
6. Fu, K., Cheng, D., Tu, Y., Zhang, L.: Credit card fraud detection using convolutional neural networks. In: International Conference on Neural Information Processing. pp. 483–490 (2016)
7. Jiang, C., Song, J., Liu, G., Zheng, L., Luan, W.: Credit card fraud detection: A novel approach using aggregation strategy and feedback mechanism. IEEE Internet of Things Journal (2018)
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436 (2015)
9. Liaw, A., Wiener, M., et al.: Classification and regression by randomforest. R news **2**(3), 18–22 (2002)
10. Malekian, D., Hashemi, M.R.: An adaptive profile based fraud detection framework for handling concept drift. In: Information Security and Cryptology (ISCISC), 2013 10th International ISC Conference on. pp. 1–6. IEEE (2013)
11. Masud, M., Gao, J., Khan, L., Han, J., Thuraisingham, B.M.: Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Transactions on Knowledge and Data Engineering **23**(6), 859–874 (2011)
12. Modi, K.: Fraud detection technique in credit card transactions using convolutional neural network (2017)
13. NilsonReport: The nilson report (Oct 2016), https://www.nilsonreport.com/upload/content_promo/The_Nilson_Report_10-17-2016.pdf
14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
15. Pozzolo, A.D., Caelen, O., Borgne, Y.A.L., Waterschoot, S., Bontempi, G.: Learned lessons in credit card fraud detection from a practitioner perspective. Expert Systems with Applications **41**(10), 4915–4928 (2014)
16. Srivastava, A., Kundu, A., Sural, S., Majumdar, A.: Credit card fraud detection using hidden markov model. IEEE Transactions on Dependable and Secure Computing **5**(1), 37–48 (2008)
17. Wang, P., Zhang, P., Guo, L.: Mining multi-label data streams using ensemble-based active learning. In: Proceedings of the 2012 SIAM international conference on data mining. pp. 1131–1140. SIAM (2012)
18. Whitrow, C., Hand, D.J., Juszczak, P., Weston, D., Adams, N.M.: Transaction aggregation as a strategy for credit card fraud detection. Data Mining and Knowledge Discovery **18**(1), 30–55 (2009)