IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 July 2022, accepted 5 August 2022, date of publication 17 August 2022, date of current version 22 August 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3199425*

RESEARCH ARTICLE

# Feature Engineering and Resampling Strategies for Fund Transfer Fraud With Limited Transaction Data and a Time-Inhomogeneous Modi Operandi

**YU-YEN HSIN[1], TIAN-SHYR DAI[2,3], YEN-WU TI[4], MING-CHUAN HUANG[5], TING-HUI CHIANG[6], AND LIANG-CHIH LIU[7]**

[1]Institute of Finance, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan
[2]Department of Information Management and Finance, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan
[3]Risk and Insurance Research Center, National Chengchi University, Taipei 116, Taiwan
[4]College of Artificial Intelligence, Yango University, Fuzhou, Fujian 350015, China
[5]Institute of Computer Science and Engineering, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan
[6]Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan
[7]Department of Information and Finance Management, National Taipei University of Technology, Taipei 106, Taiwan

Corresponding author: Tian-Shyr Dai (cameldai@nycu.edu.tw)

The work of Tian-Shyr Dai and Ting-Hui Chiang was supported in part by the Ministry of Science and Technology (MOST), Taiwan, under Grant 110-2634-F-A49-004-. The work of Yen-Wu Ti was supported in part by the Major Project Funding for Social Science Research Base in Fujian Province Social Science Planning under Grant FJ2020MJDZ055.

**ABSTRACT** Detecting financial fraud to profile crimes and pinpoint system vulnerabilities is an essential issue in the financial industry. Because of interpretability requirements and the lack of mass transaction data due to privacy regulations, sophisticated handcrafted features have been adopted in much of the literature for fraud detection. In addition to established recency, frequency, monetary, and anomaly features, we propose behavior- and segmentation-type features based on statistical characteristics belonging solely to (non-)fraudulent accounts informed by financial expertise. Our proposed features are difficult for automatic feature generators to synthesize, and provide transparent cause-effect relationships and good prediction results. Features with time-inhomogeneous properties cause popular boosting classifiers such as XGBoost and LGBM to produce unstable detection results. We use the Kolmogorov–Smirnov test to detect and remove these features to improve XGBoost and LGBM detection performance and robustness. The resulting performance shown in our experiments is better than that of other classifiers, such as SVM and random forests. We examine the advantage of our technique by comparing it with several feature engineering works on fraud detection and automatic feature generation methods. On the other hand, we also find that generating training/testing sets with random sampling falsely eliminates such time inhomogeneity and results in misleading assessments of the robustness of machine learning models. These time-inhomogeneous phenomena also entail various modus operandi patterns, which influence the performance of different resampling methods for addressing data imbalance in fraud detection. Improper linear interpolation of SMOTE-related approaches leads to poor performance due to varying patterns of modi operandi. However, synthesizing fraudulent samples with simple oversampling and GANs mitigates this problem.

**INDEX TERMS** Electronic fund transfer fraud detection, feature engineering, Kolmogorov-Smirnov test, resampling, feature importance ranking.

## I. INTRODUCTION

With the emergence of new information technologies and the evolution of various financial services, the magnitude and variety of financial fraud have also grown. Common financial frauds include credit card fraud, fund transfer fraud, insurance fraud, mobile communication fraud, etc. Such frauds lead to considerable economic losses and therefore incur high fraud detection, management, and law enforcement costs. For example, a malicious scammer could guide victims to transfer

The associate editor coordinating the review of this manuscript and approving it for publication was Yongming Li.

money from their accounts to a criminal gang's account via phone or social communication platforms, such as Facebook and Line, and thereby commit a fund transfer fraud. In fact, the total loss of fraud in 2019 was 28.3 billion USD with very low clear-up rates as reported by the Communication Fraud Control Association.[1] Fund transfer fraud (FTF), such as romance scams, buyer overpays, etc.,[2] is difficult to prevent and detect. Various fraud prevention acts have been enacted throughout the world [2], including (in Taiwan) the Proceeds of Crime Act, the Money Laundering Prevention Act, and the Money Laundering Control Act.

Financial institutions must follow fraud prevention guidelines to detect crime, profile modi operandi, and identify vulnerabilities in fund transfer systems. Rule-based models are still widely adopted by Taiwan's financial institutions to identify suspicious accounts, but they generally fail to recognize the complicated and time-varying characteristics of modi operandi (i.e., the dataset shift problem [1]) committed by fraudulent actors [3]. For example, the rule-based model provided by our partner bank (Bank L hereafter) produces an extremely low recall rate (5.56%) with a poor precision rate (40%). This incurs huge management costs and infringes on normal users' rights to access financial services without actually preventing fraud. Thus, it is critical to develop a high-performance fraud detection system with fair interpretability.

Recent studies broadly apply machine learning to detect FTFs. Although adopting automatic feature synthesis from raw data has recently become popular, the lack of sufficient transaction data limits its performance. Indeed, the amounts of raw transaction data available for training a machine learning model are limited due to privacy regulations. Whitrow *et al.* [4] also point out the difficulty of feature synthesis due to the high-dimensional nature of raw transaction data. In addition, automatic feature synthesis usually does not satisfy interpretability requirements and may require much running time, as verified in our experiments. However, sensible reasons (generated from interpretable features) are required to screen or freeze suspicious accounts. To address the above issue, Bhattacharyya *et al.* [5], Bahnsen *et al.* [6], and Whitrow *et al.* [4] study handcrafted features to retrieve patterns from raw transaction data. Baesens *et al.* [7], Zhang *et al.* [8], Xie *et al.* [9], and Bahnsen *et al.* [6] collect the features generated by previous fraud detection studies. Then they group these features into recency, frequency, money, and (unsupervised) anomaly detection (abbreviated as RFMA). Most of these features are constructed based on mathematical or statistical properties and involve little financial expertise. Baesens et al. [7] point out that data engineering is of the utmost importance to improve fraud detection performance even with simple machine learning models. In light of the above observations, we extend our conference work [10] to develop two new feature categories, namely, behavior and segmentation — generated based on characteristics belonging solely to (non-)fraudulent accounts informed by financial expertise. Such feature constructions can capture cause-effect relationships between modi operandi and features to improve interpretability; our later experiments show that the importance of these features ranks high, which reflects their strong relationship. Behavioral features capture critical transaction patterns that are typically used only by fraudsters (or normal users) and that cannot be properly captured by the construction guidelines of recency, frequency, monetary, and anomaly raised in the literature. For example, a fraudster seeks to withdraw as much as possible before the freezing of the fraudulent account, and thus the amount attempted to be withdrawn from an account is often larger than the account balance. Such features are generally constructed based on financial expertise, which can be interpreted as the knowledge base of an expert system. This reveals that combining the last-generation AI (i.e., expert systems [11]) and the current-generation AI can produce better classification results given limited amounts of training data. Segmentation features are constructed by dividing the raw transaction data according to classification rules and then compiling summary statistics to extract meaningful phenomena. For example, we first calculate the number of fraudulent accesses to each ATM and note its owner bank. Next, banks can be classified by the total number of fraudulent ATM accesses; this is because the management and location selection strategies of a bank may affect the likelihood of fraudulent access. In addition to analyzing fraudulent behaviors, we follow the idea of Abdallah *et al.* [12], who use features to capture the patterns of non-fraudulent behaviors. This indirectly improves fraud detection performance by strengthening the ability to recognize normal accounts. For instance, a user can assess over-the-counter services only by being physically present at the bank branch; fraudsters are unlikely to assess such services to avoid exposing their identities. Then a new feature is created by analyzing 405 different types of transactions to determine those that are used almost exclusively by normal users.

To analyze the performance of our proposed features, we first compare the performance of some feature engineering works on fraud detection with our real transaction data from Bank L. However, many proposed features in their works depend on the specific properties of their data and are difficult to apply to our data. This might explain the poor performance in training with their feature sets. To make the comparison fair, we collect and implement features from fraud detection feature engineering works, such as [2], [4], [5], [6], [7], [9], [13], [14], [15], [16], [17], [18], and [19]. Then, we train with the features collected from past research and (or) behavior- and segmentation-type features to compare the fraud detection performance of different classifiers, such as support vector machines (SVMs), random forests, and extensions of gradient boost decision trees like XGBoost and light gradient boosting machines (LGBM). Although XGBoost and LGBM

[1] See https://cfca.org/wp-content/uploads/2021/02/CFCA-2019-Fraud-Loss-Survey.pdf

[2] See https://www.worldremit.com/en/stories/story/2020/01/20/money-transfer-scams

have the potential to yield the best detection results due to their sophisticated boosting techniques, incorporating noisy features could significantly deteriorate the prediction performance. Specifically, the distributions of noisy features change significantly with time, making a machine learning model designed to fit the training set fail to detect fraud effectively in the testing set. To identify features with time-inhomogeneous characteristics, we propose a new scheme to measure the difference of a feature's distributions in the training and testing sets split by chronological order in terms of $p$-values generated by the Kolmogorov–Smirnov test (the KS test).[3] The aforementioned deterioration in recall and precision rates can be addressed by removing features whose distributions vary significantly (i.e., have low $p$-values). In addition, training XGBoost/LGBM with only our features and with our features in addition to those in previous works (except for noisy features) yields comparable detection performance; however, using our feature set requires less running time. Note that generating training/testing sets via random sampling can mistakenly improve fraud detection performance significantly because time-inhomogeneous characteristics in real data are eliminated. Besides, such unstable prediction problems do not occur in SVMs and random forests, possibly because sophisticated boosting techniques are not used in these classifiers. In this case, the detection performance improves as features are added, even for noisy features.

In addition to the handcrafted feature generation mentioned above, automated feature engineering is used in Kanter and Veeramachaneni [21], Lucas *et al.* [18], Esenogho *et al.* [22], and Ebiaredoh-Mienye *et al.* [23] to detect fraud. Kanter and Veeramachaneni [21] use a deep feature synthesis algorithm and substitute raw data via a transform primitive to generate primitive features. These features are then substituted into aggregation primitives that use relations between database tables to store different types of data elements to systematically generate aggregated features. Lucas *et al.* [18] use multi-perspective hidden Markov models to examine the amounts and recency of transaction sequences from the histories of credit card holders/merchants. Esenogho *et al.* [22] use long short-term memory (LSTM) to capture temporal patterns from credit card transaction data. Then, they train an adaptive boosting model with these synthetic features to detect credit card fraud. Ebiaredoh-Mienye *et al.* [23] use the stacked sparse encoder to generate feature representation for each observation for predicting credit card defaults. We train with features generated by the above automatic feature generation models, but most detection results are poor. One possible reason is that most fraud detection works study credit card frauds, which provide more aspects and larger amounts of transaction information than bank accounts' fund transfer data. Another reason might be that existing automatic feature generators can generate simple features, such as RFMA proposed in past literature, but find it challenging to create sophisticated behavior or segmentation features that require complex generation procedures and financial expertise. Our feature constructions, in contrast, can provide new insight into automatic feature generation given limited amounts of training data. In addition to extracting RFMA features from raw transaction data, generating a statistical summary of labels (i.e., fraudulent accesses) for raw transaction data with exogenously-obtained categorization information could yield useful features. For example, as mentioned above, we classify ATMs according to their owner banks and then count the number of fraudulent accesses for each bank to identify suspicious banks, as in Table 4. Similarly, statistical summaries of fraudulent/normal accesses classified by ATM branch (location) and transaction type also produce useful features as detailed in Figure 2 and Table 3, respectively.

The ratio of the number of fraudulent accounts to normal accounts is usually extremely biased, namely, $1:250$ in our dataset. Such data imbalance can cause machine learning models to predict all account observations as normal. However, it is generally more important to identify fraudulent accounts (i.e., improve the recall rate) than to achieve high prediction accuracy. Accordingly, we resample the training dataset to balance the ratio of positive/negative observations. We find that modus operandi patterns can be divided into several subgroups as observed from a scatter plot projected from high-dimensional feature vectors of accounts. Thus, even through Vassallo *et al.* [27] suggest that SMOTE-NCL is especially useful for dealing with financial data imbalance, our experiments show that SMOTE-related approaches yield degraded prediction for time-inhomogeneous fraud detection. This is because interpolations adopted by SMOTE-related methods can improperly place synthesized positive observations where negative observations are dense and/or change the statistical properties of features belonging to positive observations. This problem can be avoided by adopting naive full/random oversampling methods or Wasserstein generative adversarial networks (GANs) (see [24]). Our experiments show that GANs outperform oversampling methods, which in turn outperform SMOTE-related methods.

In addition to predicting modi operandi, the ability to interpret the predictions for summing up the patterns of modi operandi is also essential for an FTF detection system to pinpoint the vulnerabilities of the procedures of financial services. Indeed, profiling modi operandi prevents fraudsters from utilizing fund transfer systems (see [25]) and fulfilling the "risk-oriented"[4] property asked for by the Financial Action Task Force[5] (see [26] and [27]). Our proposed behavioral and segmentation features provide a clear cause-effect relationship between features and fraudulent labels. Their excellent qualities can be examined by showing that our features generally rank high under the feature importance

[3] The KS test is also used to remove features that have the same distributions in the positive and negative observations as in [20]; note that this idea is quite different from ours.

[4] Efforts should be allocated where the fraudulent risk is likely.

[5] https://www.fatf-gafi.org/

ranking procedure proposed in [28] published in a core finance journal (*Journal of Banking and Finance*.)

We organize the structure of the remaining paper as follows. We first survey relevant fraud detection studies in Section II. Section III describes the data format provided by the partner bank. Section IV describes our feature engineering approach. The experiments in Section V are divided into three parts: Section V-A compares the performance of various machine learning models with the features suggested in the literature and those in this paper. We also analyze how the time inhomogeneity of input features influences the stability of fraud detection. Section V-B addresses the data imbalance by comparing the performance of various resampling methods and analyzes their relationship to the distributions of original/synthetic observations. Section V-C analyzes feature importance, and Section VI concludes.

## II. LITERATURE REVIEWS

Financial fraud includes credit card fraud [17], [19], [29], phone fraud [30], online transaction fraud [31], instant payment fraud [32], etc. To ensure the interpretability of the detection results, most conventional banks detect frauds by rule-based methods. However, these methods generally fail to capture complex, time-varying characteristics of modi operandi; fraud detection performance thus tends to be poor. In addition, fraudsters can easily bypass these fixed rules.

It is impractical to train a machine learning model with raw transaction data due to data availability constrained by regulations and the heterogeneous and high-dimensional nature of transactions. Whitrow *et al.* [4] and Bhattacharyya *et al.* [5] train machine learning models with features extracted from aggregated raw transaction data to detect fraud. This process generates a set of features to capture insightful properties of (non-)fraudulent accounts. Xie *et al.* [9] show that many features generated in previous studies are generally based on transaction frequency. However, capturing temporal properties without considering the characteristics of modi operandi from other aspects hinders machine learning models from recognizing a wider variety of fraudulent behaviors. For example, chronological relationships in raw transactions are difficult to capture by frequency features. We note in this connection that fraudsters usually first do transactions with small amounts to test the vulnerability of a transaction system, and then do a large one. Xie *et al.* [9] argue that using interpretable monetary features to capture patterns from transactions of fraudulent accounts dramatically enhances fraud detection results. Zhang *et al.* [8] suggest that the features generated by recent fraud detection studies can be categorized into recency, frequency, and monetary (RFM) groups. In addition to RFM, Baesens *et al.* [7] propose another two groups: anomaly detection and other feature engineering techniques. They empirically compare the fraud detection results by training with the above features and show that excellent performance can be achieved by adequately constructing feature sets without using sophisticated machine learning models. In light of the above research, we create new features by carefully observing (non-)fraudulent behaviors and using financial expertise to capture their patterns to propose two feature categories: behavior- and segmentation-type features.

Advanced machine learning models such as graph-based models are utilized due to the availability of certain types of transaction information. Wang *et al.* [32] build a heterogeneous attribute graph to represent accounts' social relations and frequently used locations of online merchants by using semi-supervised graph embeddings to produce a low-dimensional representation for each node. To detect instant payment fraud with interpretable results, they use a hierarchical attention mechanism for each node to determine the relations between neighbors or attributes. Li *et al.* [33] identify paths of fraudulent fund transfers through graph-based models. Cheng *et al.* [17] extract RFM-based features from temporal and spatial information embedded in raw transaction data and detect credit card fraud by applying an attention mechanism to extract important features. Zheng *et al.* [30] aggregate transaction records (including transfer records from the sending account to the receiving account(s) and the receiving bank) from two banks to detect suspicious transfers. A GAN (generative adversarial network) is then applied with a denoising autoencoder to calculate the probability that a cross-bank transfer is fraudulent. However, such detailed spatial transaction data, social relationships, and cross-bank transaction records are unavailable in our raw transaction dataset. Hence, we do not consider such sophisticated methods.

Methods for detecting anomalies like fraudulent accounts generally face significant data imbalance problems; that is, the distribution of the training dataset is biased, with few/many observations in minority/majority classes. To mitigate the resulting learning bias toward majority classes, resampling procedures are used to alter the ratio of positive to negative observations (usually to be closer to 1) by oversampling, undersampling, or hybrid methods as categorized in [34]. Oversampling methods increase the number of observations in the minority group (e.g., fraudulent accounts in our case) by replicating or synthesizing new ones. Undersampling methods filter out observations from the majority group (e.g., nonfraudulent accounts). Hybrid methods combine oversampling and undersampling. Ghorbani and Ghousi [34] and Hordri *et al.* [35] compare different resampling methods, including SMOTE, borderline SMOTE, SMOTE-ENN, SVM-SMOTE, SMOTE-Tomek, and random under/oversampling. Vassallo *et al.* [27] claim that SMOTE-NCL is especially useful for dealing with financial data imbalance. We compare the above methods in our experiments, and consider the Wasserstein GANs [24].

Addressing imbalanced binary classification problems, such as fraud detection using machine learning methods, has also been a recent focus in academic financial journals. Khandani *et al.* [36] use generalized regression and classification trees to predict the delinquency and default rates of credit card holders since interpretable decision logic can be obtained from the rules of each tree node and the tree structure. They demonstrate significantly improved detection rates and show that such interpretable analyses may have important applications in forecasting systemic risk, that is, the risk of major collapses, such as the financial crisis of 2007–2008. Butaru *et al.* [28] also show that decision trees and random forests perform well in predicting credit card delinquency for different banks. However, they criticize the poor interpretability of these tree-based models, as the features selected by these models vary with time and banks. Moreover, the complex structure and high number of leaves of the trees complicate the comparison of overall feature selection results. To profile and compare the results of feature selection, they measure and rank the importance of each feature by the number of occurrences, the occurrence position (in the tree), and the information gain of each feature. This measures the effectiveness of each feature and further improves the interpretability and cause-and-effect of their machine learning models. In Section V-C, we adopt the ranking method of [28] to show the effectiveness of our proposed features based on XGBoost. Addo *et al.* [37] build binary classifiers based on different machine learning models to predict the probability of loan default. They rank the ten most important features and use them to assess the stability and performance of prediction among different models. They show that tree-based models outperform other models. Zhang *et al.* [38] compare major machine learning algorithms and sampling techniques to detect money laundering. They argue that decision trees are more flexible than parametric methods like logistic regression in capturing nonlinearity and accounting for missing values and outliers. However, decision trees suffer from overfitting and thus require stopping rules, for instance, to limit the maximum depth of the tree or the number of branches.

## III. DESCRIPTIONS AND PREPROCESSING OF RAW TRANSACTION DATA

Bank L has provided us with raw transaction data from April 2018 to September 2019 and the records of fraudulent accounts provided by the National Police Agency. This paper proposes sophisticated features extracted from real raw transaction data that contain rich details of various patterns of normal and fraudulent behaviors, as shown in Table 1. Although Kaggle provides public datasets for fraudulent detection,[6] the datasets are either generated by simulations or provide limited disclosure information due to strict regulations of privacy. These drawbacks limit the ability to profile the characteristics of normal and fraudulent transaction behaviors by generating interpretable features. Chen *et al.* [39] show that due to this limitation, anti-fraud works such as [40] and [41] merely consist of descriptions of methods without experiments. Hence, our studies focus on real transaction data provided by Bank L. Inputting the entire transaction record to train a machine learning model to detect fraudulent transactions is unrealistic (see [4]) due to the high-dimensional nature and heterogeneity of the raw transaction data. Thus, the feature representation of each bank account is constructed by first aggregating $n$ transactions for that account that occurred within a predetermined period. Then these studies extract features from aggregated transactions. This aggregate approach is widely adopted in [2], [4], [5], [6], [7], [9], [14], [15], [16], [17], [18], and [19]. However, determining the hyperparameter $n$ faces a dilemma since increasing (decreasing) $n$ results in more (fewer) aggregated transactions to describe the characteristics of a bank account, but fewer (more) bank accounts being included in the training/test dataset. Note that many accounts are used infrequently and are removed when we set a large $n$. Such removals may significantly damage the training of fraud detection systems because fraudulent accounts are rare and many of them are infrequently used. To preserve a fair percentage of fraudulent accounts for training without sacrificing the number of aggregated transactions, we set $n$ to 9. As observed in Fig. 1, we observe that the ratio of fraudulent accounts used for training drops rapidly when the hyperparameter $n$ is larger than 9. The percentage of aggregated fraudulent transactions to all transactions of fraudulent accounts is also higher for the $n = 9$ scenario.

**TABLE 1. Transaction data descriptions.**

| Item | Description |
|---|---|
| Account No. | Each account is identified by a unique identification number. |
| Transaction date | Date of transaction |
| Transaction type | One of 405 transaction types (in code), such as a deposit, a withdrawal, and an inter-bank fund transfer. |
| Debit amount | The amount of money paid out of the account, such as withdrawal. The field is empty if not applicable. |
| Credit amount | The amount of money paid into the account, such as deposit. The field is empty if not applicable. |
| Balance amount | The amount of money owed by an account. |
| Note | Relevant information about transaction details, such as the serial number of the ATM for the transaction |
| E-bank | Indicating whether E-bank services are involved in the transaction |
| Telephony | Indicating whether telephony services are involved in the transaction |
| Fraudulent account | Whether the account is alerted by the National Police Agency |

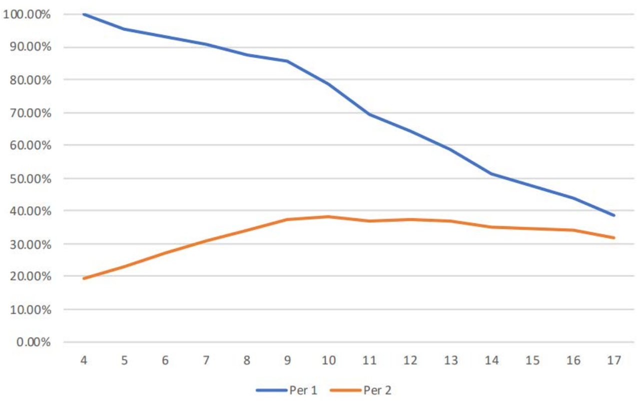


**FIGURE 1. Ratio of fraudulent accounts and aggregated transactions used in training/testing datasets Ratio1: Ratio of fraudulent accounts executing more than *n* transactions within the predetermined period Ratio2: Ratio of aggregated fraudulent transactions used in training/testing datasets to all transactions belonging to fraudulent accounts.**

[6] Like https://www.kaggle.com/ealaxi/paysim1 and https://www.kaggle.com/mlg-ulb/creditcardfraud/home

## IV. CONSTRUCTION OF BEHAVIOR AND SEGMENTATION-TYPE FEATURES

Whitrow *et al.* [4] and Bhattacharyya *et al.* [5] retrieve the patterns of normal and fraudulent transactions by generating the features from aggregated transactions. These features are then used to train classification models to predict fraudulent accounts. Much of the literature follows this approach, and the features considered in the literature can be categorized as recency, frequency, money, and anomaly detection techniques (see [7]). This paper extends our previous conference work [10] to collect the features that can be generated based on our raw transaction data[7] from [2], [4], [5], [6], [7], [9], [14], [15], [16], [17], [18], and [19], as illustrated in the upper part of Table 2. For ease of analyzing their effectiveness in fraud detection, the set of all these features is denoted as **Other** in the following experiments. Along with the newly proposed features, we construct two novel feature categories: behavior and segmentation, and denote the resulting feature set as **Proposed**. The **Proposed** categories are illustrated in the lower part of Table 2 and the feature constructions from **Proposed** are discussed in the following two subsections. Our experiments show that training gradient-boosting decision trees with the proposed features could yield good fraud detection results. This echos the findings in [7] that careful feature engineering without sophisticated machine learning techniques can also achieve good performance.

Generally speaking, the literature on past fraudulent detection creates features mainly based on abnormal patterns of modi operandi for identifying fraudulent accounts. Note that fraudsters must execute illegal behaviors to commit crimes even though their behaviors are generally similar to those of normal account holders most of the time in order to cover their identity. Thus, it is straightforward to identify fraudsters to pinpoint ‘‘what fraudsters do’’. However, we find that profiling certain normal behaviors avoided by fraudsters is also beneficial. This is because fraudsters avoid certain behaviors to prevent increasing the risk of disclosing their identity or disturbances to their criminal plans. Accounting for both ‘‘what fraudsters do not do’’ and ‘‘what fraudsters do’’ helps distinguish fraudsters from normal users with similar transaction characteristics. Indeed, our improvements would reduce the number of false alarms, hence the inconvenience for normal account holders and the associated labor costs to screen or freeze the accounts [6]. Since the features constructed according to the guidelines of ‘‘what fraudsters (don’t) do’’ cannot easily fit into RFMA proposed by Baesens *et al.* [7], we add the categories ‘‘behavior’’ and ‘‘segmentation.’’

### A. BEHAVIOR FEATURES

Features belonging to the behavior category profile characteristics of (non-)fraudulent transactions. These features cannot be easily categorized into RFMA but are critical to recognizing the modus operandi patterns of financial expertise. We list the behavior-type features as follows.

#### 1) IMMEDIATE_PAID_OUT

We count the occurrences of payment made from an account immediately following the event of payment made into the same account on the same day. Note that fraudsters want to transfer funds before the police investigate and freeze the account in question.

#### 2) ATM_TRANSACTION

An ATM transaction is one of 405 types of transactions. We focus on this transaction type since most FTFs utilize ATMs to transfer or dispatch money, as it is convenient, safe, and unlikely to disclose the identities of fraudsters. We count the number of transactions using ATMs from all $n$ aggregated transactions of the same account defined in Figure 1. Note that the 405 transaction types cause the one-hot encoding to result in unnecessarily high-dimensional inputs and poor fraud detection performance. Additionally, there is no natural order for these 405 transaction types, making label encoding unreasonable. In addition, target encoding is impractical since the number of fraudulent accounts to normal ones is extremely biased. Thus, we propose the following features to aggregate information on transaction types.

#### 3) LT_COUNT

We count the occurrences of ‘‘likely legal’’ transactions, whose transaction types are frequently conducted by normal users but typically avoided by fraudsters, from aggregated transactions of an account. Some transaction types, such as withdrawals or deposits on bank counters, increase the risk of being identified or getting caught. Other types, such as purchasing and redeeming investment products, are generally irrelevant to the modi operandi. The five most frequently used transaction types used by (non-)fraudulent accounts are illustrated in Table 3; three of these are common to both normal and fraudulent accounts. This shows that naively identifying high-frequency transactions used by (non-)fraudulent accounts does not yield useful features. Our sophisticated feature generation idea is useful since LT_count is found to be the second most important feature, as will be discussed later in Table 13.

#### 4) LAST_PAID_OUT_LARGER_THAN_SAVINGS

This denotes a case in which the last amount paid out from the account was larger than the balance of that account. Note that fraudsters would try their best to transfer funds from fraudulent accounts before these accounts are frozen.

#### 5) FRAUD_FACTOR

It represents the likelihood of fraud as the product of several fraud-related features:

$$\begin{aligned}&(\text{ Last\_paid\_out\_larger\_than\_savings })/n\\ &\times(\text{ Immediate\_paid\_out })/n\\ &\times(\text{ ATM\_transaction })/n,\end{aligned}$$

[7] We ignore features such as the locations of transactions that are not contained in our raw transaction data defined in Table 1. The definitions of the features used in our experiments can be found in Table 4 of [10].

**TABLE 2. Feature categories. Numbers in parentheses denote feature importance rankings proposed by Butaru *et al.* [28] that will be discussed later in Table 13. "*" in parentheses denotes a feature importance of zero. We use red (black) color or "Proposed" ("Other") to denote features used only in this study (in past literature categorized by [7]).**

| Feature categories from Baesens et al. (2021) | | | |
|---|---|---|---|
| **Recency** | **Frequency** | **Money** | **Features based on (unsupervised) anomaly detection techniques** |
| 24hr_transaction (16) | Average_transaction_interval (4) | Withdrawal_stdev (11) | Mahalanobis_anomaly (35) |
| Amount_over_month (36) | Least_frequent_transaction_type (8) | Deposit_stdev (20) | KNN_distance (*) |
| Average_daily_over_month (39) | Most_frequent_transaction_type (3) | Suspicious_amount_count (15) | LOF (*) |
| Amount_transaction_object_over_month (23) | Second_most_frequent_transaction_type (6) | Total amount (30) | Isolation_tree (14) |
| Average_over_2_months (26) | Note_not_empty_count (17) | Consecutive_transact_type_amount (10) | Zscore_outlier_count(*) |
| Amount_transaction_object_over_2_months (22) | Consecutive_transact_type_count (28) | MAD (19) | Fits_Benford (38) |
| | Untrusted_frequent_trade_count (5) | Zero_digit_freq (34) | |
| | | Max_amount_same_day (12) | |
| | | Max_number_same_day (9) | |
| | | Sensitive_single_amount_count (33) | |
| | | Sensitive_daily_total_amount_count (37) | |
| | | Sensitive_test_amount_count (25) | |
| | | Large_amount_count (40) | |
| | | Big_onetime_deal_count (7) | |
| | | Saving_gradient (13) | |

| Feature categories added in our research | |
|---|---|
| **Behavior** | **Segmentation** |
| Immediate_paid_out (24) | Suspicious_branch (32) |
| ATM_transaction (21) | Suspicious_ATM_bank (29) |
| LT_count (2) | LATM_count (31) |
| Fraud_factor (18) | Branch_ID (1) |
| Last_paid_out_larger_than_savings (27) | |

**TABLE 3. Utilization ratios and descriptions for top-5 most frequently used transaction types for normal and fraudulent accounts.**

| Top-5 types of normal accounts | | Top-5 types of fraudulent accounts | |
|---|---|---|---|
| Ratio | Description | Ratio | Description |
| 14.3% | Cross-bank withdrawal | 14.3% | Cross-bank withdrawal |
| 11.1% | Withdrawal using ATM card | 11.3% | Withdrawal using ATM card |
| 8.1% | Transfer of term deposit interests | 7.7% | Cross-bank transfer |
| 7.6% | Cross-bank transfer | 3.1% | (Another account performs) at-the-counter transfer to this account |
| 5.2% | At-the-counter transfer (to another account) | 3.0% | Processing fee |

where $n$ denotes the feasible number of aggregated transactions determined in Figure 1. This feature facilitates the capture of coexisting occurrences of fraud-related features to measure the likelihood of fraud.

## B. SEGMENTATION FEATURES

Segmentation features are constructed by discovering useful classification rules with summary statistics. We first label each ATM machine or an account with its associated bank, branch, or other meaningful classification and then analyze their relationship to fraud. We list these features as follows:

### 1) SUSPICIOUS_ATM_BANK

We perform summary statistics for the number of fraudulent accesses to each ATM and recognize its owner bank. Then we calculate the lump sum of fraudulent ATM accesses for each bank and label those with the top 5% (see the upper panel of Table 4) of fraudulent accesses as "Suspicious ATM Banks." Note that in significance tests, five percent is a prevalent statistical threshold. High fraudulent access to ATMs belonging to a specific bank may result from the bank's ATM location selection or management policies.[8] For example, ATMs near train stations are frequently used by fraudsters.[9]

### 2) LATM_COUNT

This is the number of times an account has accessed "likely legal" ATMs, defined as ATMs that have been used by fraudulent users fewer than six times. We decided on six because the number of fraudulent accesses of 95% (a frequently used statistical threshold) of ATMs in our training set is fewer than six times (see the lower panel of Table 4).

**TABLE 4. Accumulated percentile of the number of a Bank's ATMs (an ATM) accessed by fraudulent accounts.**

| Number of times a bank's ATMs being used by fraudsters | Number of bank | Accumulated Percentile |
|---|---|---|
| 1-10 | 25 | 80.64% |
| 11-20 | 2 | 87.10% |
| 31-40 | 1 | 90.32% |
| 41-50 | 1 | 93.55% |
| 51-60 (Suspicious ATM Banks) | 1 | 96.77% |
| 131-149 (Suspicious ATM Banks) | 1 | 100% |
| **Number of times an ATM being used by fraudsters** | **Number of ATM** | **Accumulated Percentile** |
| 1-4 (Likely legal ATM) | 342 | 88.60% |
| 5 (Likely legal ATM) | 20 | 93.78% |
| 6-7 | 21 | 99.22% |
| 8-10 | 1 | 99.48% |
| 11-12 | 1 | 99.74% |
| 41-44 | 1 | 100% |

### 3) SUSPICIOUS_BRANCH

This specifies whether ATM transactions in an account are executed where a lot of fraud occurs. Although actual ATM locations cannot be extracted from the raw transaction data (see Table 1), we instead identify each ATM's owner branch by comparing the serial number of the bank branch with the serial numbers of its ATMs. Then we profile the area where a branch office is located with its ATM transaction data since ATMs owned by the same branch office are located close to the office. We label the branches owning ATMs that have been accessed only by fraudsters as suspicious branches, as illustrated in Figure 2. Note that it does not imply that ATMs of suspicious branches are accessed only by fraudsters, as we have access to the transaction data of only Bank L accounts within a limited period.

### 4) BRANCH_ID

This summarizes the branch to which (non-)fraudulent accounts belong; it can also be used to generate the above feature.

[8] For instance, a bank may deploy ATMs in branches of a chain store with which it cooperates.

[9] See the news https://news.tvbs.com.tw/local/1415294

Recently, studies of open-source tools for automatic feature synthesis have become popular, such as Featuretools. This tool extracts features from raw data via primitives such as averages, sums, minima, maxima, standard derivations, and the skew of raw data. The aggregates of the primitives of the raw data are then used to synthesize features, such as average debit amounts and maximum account balance. Using these primitive aggregation techniques is challenging to generate behavioral- and segmentation-type features. Nevertheless, our discussion of feature construction suggests a new feature synthesis strategy. Specifically, many of the above features are constructed by forming a statistical summary of raw transaction data based on the labels (of normal and fraudulent accounts) and exogenously obtained classification information. For example, we can obtain ATM sequence numbers only from raw transaction data. By labeling each ATM sequence number with exogenous classification information, such as the bank or branch location it belongs to, we can construct a statistical summary as in Table 4. Using fraudulent and normal access labels, we can form features such as Suspicious_ATM_bank and Suspicious_branch to mark banks or branches with many fraudulent accesses. Given the wide variety of exogenous classification information related to different attributes of raw transaction data, it is not efficient to manually collect this information to construct statistical summaries and features. Our findings provide a possible development path for automatic feature construction: various segmentation features could be efficiently generated by using spiders to extract useful classification information from the Internet in combination with a statistical summary generator.

The following experiments compare fraud detection performance using behavior and segmentation features analyzed above with RFMA features and automatic feature engineering methods proposed in previous work. We also analyze the characteristics of noisy features that degrade detection performance and examine the performance improvements obtained by removing these features. The quality and interpretability of our proposed features are attested by their high ranking in terms of the feature importance proposed in [28] published in a premium finance journal.

## V. EXPERIMENTS

Note that banks are required to provide qualified and interpretable fraudulent detection systems. To echo these requirements, we compare fraudulent detection performance under different feature engineering and resampling techniques in Sections V-A and V-B. Feature importance rankings in Section V-C analyze the interpretability of the features proposed in our paper and previous works.

### A. ANALYSES OF FEATURE ENGINEERING TECHNIQUES

We first compare the fraud detection performance by training popular classifiers, such as SVM, random forests, XGBoost, and light GBM (LGBM), with different feature engineering models and Bank L's real transaction data. We collect our proposed features and RFMA features surveyed in the past literature as defined in Table 2 and train different combinations of the features to compare the fraud detection performance. We also study unstable fraudulent detection results caused by time-inhomogeneous features and remove these features using the Kolmogorov-Smirnov test. In addition, the performance of automatic feature synthesis algorithms is also compared in this section.

We sort all transaction accounts and their aggregated transaction information in chronological order. The first 60% (last 40%) of the data are used as the training (testing) set. We try different parameter settings to optimize the fraudulent detection performance, and the best settings in our experiments for each machine learning model are shown in Table 5. As a severe data imbalance hinders the recognition of the characteristics of minority samples (i.e., fraudulent accounts), we use full oversampling to adjust the ratio of fraudulent to non-fraudulent accounts to 1 : 1, where all minority observations are duplicated an equal number of times. The effects of other resampling methods will be studied in Section V-B. Then we use features proposed by credit card fraud detection

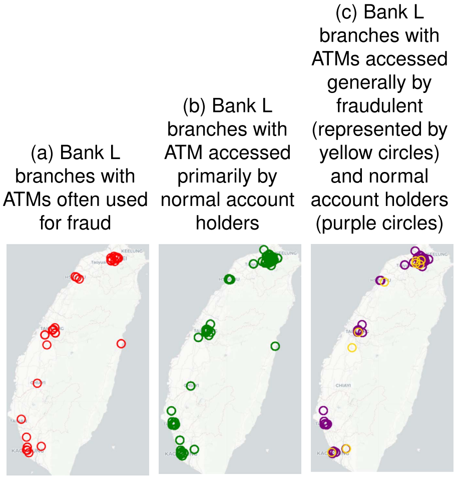

(a) Bank L branches with ATMs often used for fraud
(b) Bank L branches with ATM accessed primarily by normal account holders
(c) Bank L branches with ATMs accessed generally by fraudulent (represented by yellow circles) and normal account holders (purple circles)

**FIGURE 2. Normal and fraudulent accesses of ATMs and locations of their owner branches.**

**TABLE 5. Parameters of each machine learning model. The specification of the computer: Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz with 32GB RAM and RTX3070 graphics card.**

| model | parameter |
|---|---|
| XGBoost | colsample_bytree=0.8, importance_type='gain', learning_rate=0.1, max_depth=4, min_child_weight=4, n_estimators=300, subsample=0.8, tree_method='exact', validate_parameters=1 |
| Random Forest | n_estimators = 100, criterion = 'gini', max_depth = 2, min_sample_split = 2, min_samples_leaf = 1 |
| SVM | C=1.0, kernel = 'rbf', degree = 3, gamma = 'auto' |
| LGBM | objective = regression, boosting = gbdt, num_iterations = 1000, learning_rate = 0.1, num_leaves = 31 |

**TABLE 6. Comparison with features proposed by [9] and [7].**

| Feature Set | Model | Accuracy | Precision | Recall | F1- score | running time (sec) |
|---|---|---|---|---|---|---|
| [9] | XGBoost | 99.03% | 22.94% | 59.09% | 33.05% | 1.35 |
| | Random Forest | 93.13% | 4.56% | 80.30% | 8.64% | 0.53 |
| | SVM | 95.80% | 6.22% | 66.66% | 11.38% | 15.44 |
| | LGBM | 98.52% | 14.04% | 51.51% | 22.07% | 2.50 |
| [7] | XGBoost | 98.82% | 9.09% | 21.21% | 12.72% | 1.60 |
| | Random Forest | 58.77% | 0.82% | 84.84% | 1.63% | 0.61 |
| | SVM | 66.33% | 0.81% | 68.18% | 1.61% | 118.17 |
| | LGBM | 98.27% | 2.21% | 7.57% | 3.42% | 2.90 |

works [9] and [7] to predict fraudulent accounts with our Bank L's transaction data, as illustrated in Table 6. The prediction results are poor, likely because many of their features depend on specific information associated with their data and cannot be retrieved from our transaction data (see Table 1). Specifically, credit card transaction records include extensive data such as consumption locations and merchandise by which they generate meaningful features to improve credit-card fraud detection performance. In our scenario, failing to retrieve these features from Bank L's transaction dataset degrades the performance of their models. To make comparisons of feature engineering fairer, we collect all features proposed by [2], [4], [5], [6], [7], [9], [14], [15], [16], [17], [18], and [19] that can be generated based on our raw transaction data to form a feature set **Others**. Then we compared the performance by training with "Others" and (or) our proposed feature set **Proposed**, as shown in Table 7, to show the advantage of our proposed features to solve the above heterogeneity problem of raw transaction data.

Table 7 compares the performance for detecting fraudulent accounts by training different machine learning models with different combinations of features, as listed in the "Model+Data" column. Here, we first focus on gray cells that split the data chronologically. Although gradient-boosting classification models, such as LGBM and XGBoost, can achieve strong detection performance if features are properly selected, the inclusion of noisy features can generate unstable results. This echoes the finding of [42] that the classifiers mentioned above are sensitive to overfitting due to the existence of noisy data. In fact, using our ***Proposed*** features produces a good F1 score (73.95%) with a low training time; recall rates and F1 scores deteriorate significantly with high training times if we include all the features in Table 2 for training. This confirms the argument in [7], namely, that careful feature engineering improves the performance of machine learning models, as we can use far fewer features (denoted red in Table 2) to achieve good fraud prediction performance. To determine the causes of the dramatic performance drop in XGBoost and LGBM, we monitor the change in detection performance using the leave-one-out feature selection mechanism; that is, we repeatedly single out a feature for all training features. Significant performance deterioration is due to the presence of two features from the (unsupervised) anomaly detection category: LOF and KNN_distance. Simultaneously dropping these two noisy features, as shown in the right panel of Table 7, restores the XGBoost F1 scores, namely, 74.34% (w/ **Other+Proposed** w/o LOF & KNN_distance) and 68.42% (w/ **Other** w/o LOF & KNN_distance). Compared to the relatively low F1 scores of the random forest, the nonlinear kernel SVM produces slightly worse performance than XGBoost if noisy features are removed. Furthermore, SVM performance increases steadily with increasing training features without deterioration in detection ability caused by noisy features. However, a nonlinear kernel SVM cannot easily rank feature importance to capture the patterns of modi operandi or identify the weaknesses of the transaction procedures. It also takes much more running time than other models in our experiments. Under interpretability and running time concerns, the following experiments focus on improving XGBoost for simplicity.[10]

To explore why detection performance deteriorates significantly due to the presence of LOF and KNN_distance, we repeated the above experiments by sampling the training and testing sets chronologically and randomly with various proportions of the training/testing dataset, as illustrated in Table 9. We trained XGBoost with all **Other** and **Proposed** features in these experiments. The percentage in the first column denotes the proportion of data used as the training set; the remaining data were used as the testing set. Recall that each account can be represented as a vector composed of its features; KNN_distance (LOF) profiles the statistical properties of an account's overall behavior, as it describes the average distance from its feature vector to neighborhood vectors (compared to the average density around the vectors), as stated in [7]. Thus, we can determine whether the overall behavior patterns of (non-)fraudulent accounts are similar during the training and testing periods by calculating the similarity of the cumulative distribution of KNN_distance (LOF) by the KS test. The null hypothesis of the KS test is "the two distributions are the same", which we reject to adopt the alternative hypothesis—"the two distributions are different"—if the $p$ value is small.[11] The $p$ values to test the features' distributions of fraudulent and normal accounts are listed in columns 6, 7, 8, and 9.

[10] Comparisons of LGBM are ignored for simplicity since its detection performance is similar to that of XGBoost.

[11] The $p$ value is the probability of obtaining test results that are more extreme than the current result given that the null hypothesis is correct. Intuitively, the size of the $p$ value reflects the similarity of two distributions.

**TABLE 7.** Fraudulent detection performance when data are split chronologically (gray cells) and randomly (white cells). To evaluate the stability of our experiments, we conducted each experiment five times to ensure that the results do not vary significantly. The reported performance is the average of the experimental results.

| Model+Data | Accuracy | | Precision | | Recall | | F1 score | | running time (sec) |
|---|---|---|---|---|---|---|---|---|---|
| XGBoost w/ Other+Proposed | 99.60% | 99.9% | 100.00% | 98.0% | 1.49% | 76.60% | 2.94% | 86.0% | 22.03 |
| RandomForest w/ Other+Proposed | 99.20% | 97.4% | 30.12% | 12.1% | 74.63% | 89.1% | 42.92% | 21.3% | 1.44 |
| SVM w/ Others+ Proposed | 99.67% | 99.7% | 57.32% | 58.8% | 70.15% | 73.4% | 63.09% | 65.5% | 85.59 |
| LGBM w/ Others+ Proposed | 99.60% | 99.9% | 100.00% | 92.3% | 1.49% | 75% | 2.94% | 82.8% | 2.73 |
| XGBoost w/ Proposed | 99.81% | 99.8% | 84.62% | 84.8% | 65.67% | 60.9% | 73.95% | 70.9% | 6.25 |
| Random Forest w/ Proposed | 94.76% | 91.9% | 6.20% | 3.4% | 85.07% | 81.3% | 11.56% | 6.6% | 0.64 |
| SVM w/ Proposed | 98.44% | 96.9% | 16.90% | 7.9% | 73.13% | 64.1% | 27.45% | 14% | 14.23 |
| LGBM w/ Proposed | 99.78% | 99.7% | 81.63% | 75.0% | 59.70% | 51.6% | 68.97% | 61.1% | 2.38 |
| XGBoost w/ Other | 99.60% | 99.9% | 100.00% | 95.7% | 1.49% | 68.8% | 2.94% | 80.0% | 42.59 |
| RandomForest w/ Other | 97.87% | 94.2% | 12.53% | 5.5% | 71.64% | 85.9% | 21.33% | 10.4% | 1.61 |
| SVM w/ Other | 98.05% | 97.2% | 14.17% | 9.3% | 76.12% | 71.9% | 23.89% | 16.5% | 57.41 |
| LGBM w/ Other | 99.60% | 99.8% | 100.00% | 89.8% | 1.49% | 68.8% | 2.94% | 77.9% | 2.86 |

| Model+Data | Accuracy | | Precision | | Recall | | F1 score | | running time (sec) |
|---|---|---|---|---|---|---|---|---|---|
| XGBoost w/ Other+ Proposed w/o LOF&KNN_distance | 99.83% | 99.9% | 91.30% | 97.6% | 62.69% | 64.1% | 74.34% | 77.4% | 50.64 |
| RandomForest w/ Other+Proposed w/o LOF&KNN_distance | 97.09% | 96.0% | 10.71% | 7.8% | 85.07% | 85.9% | 19.03% | 14.4% | 1.61 |
| SVM w/ Other+ Proposed w/o LOF &KNN_distance | 99.67% | 99.6% | 57.32% | 47.3% | 70.15% | 67.2% | 63.09% | 55.5% | 136.36 |
| LGBM w/ Other+Proposed w/o LOF&KNN_distance | 99.77% | 99.7% | 79.59% | 68.5% | 58.21% | 57.8% | 67.24% | 62.7% | 4.52 |
| XGBoost w/ Other w/o LOF&KNN_distance | 99.78% | 99.8% | 82.98% | 84.8% | 58.21% | 60.9% | 68.42% | 70.9% | 61.67 |
| Random Forest w/ Other w/o LOF &KNN_distance | 92.13% | 93.1% | 4.41% | 4.1% | 89.55% | 85.9% | 8.40% | 8.9% | 2.21 |
| SVM w/ Other w/o LOF &KNN_distance | 96.98% | 97.5% | 9.65% | 10.2% | 77.61% | 70.3% | 17.16% | 17.8% | 69.37 |
| LGBM w/ Other w/o LOF &KNN_distance | 99.75% | 99.1% | 74.51% | 70.6% | 56.72% | 56.3% | 64.41% | 62.6% | 4.65 |

**TABLE 8.** F1 Scores when distinguishing observations from training/testing sets using discrimination learning.

| Training/Testing ratio | All Proposed + Others | Noisy features removed via KS test |
|---|---|---|
| 0.5 | 0.6930 | 0.5042 |
| 0.6 | 0.8090 | 0.7367 |
| 0.7 | 0.8459 | 0.8222 |
| 0.8 | 0.8884 | 0.8661 |

In Table 9, we observe that chronological sampling (gray cells) yields highly divergent recall rates (1.49%–70.59%) and F1 scores (2.94%–82.76%) with changes in the training data size. These values seem to be highly correlated to the similarity of the cumulative distributions of KNN_distance and LOF proxied by the $p$ values of the KS test. Specifically, low $p$ values of these two features of fraudulent accounts also map to low recall rates, that is, the likelihood of detecting fraudulent accounts. Recall rates increase (70.59%) when $p$ values increase. These relatively low $p$ values are evidence that the modi operandi vary widely; therefore, the patterns captured from the training set may become invalid in the testing set, resulting in low and unstable recall rates. However, the $p$ values of normal accounts are higher and more stable, which suggests relatively stable behavior for normal accounts and hence high precision rates (87.5%–100%). Note that the relatively low precision of 87.5% also maps to a low $p$ value of 0.3% given a training set composed of 70% of the data.

The time inhomogeneous properties of the KNN_distance and LOF features of fraudulent accounts disappear if the training/testing sets are partitioned by random sampling (white cells). As shown in Table 9, the $p$ values in the chronological sampling cases are generally lower than those in the random sampling cases. This is because generating training and testing datasets by randomly sampling the raw transaction dataset results in similar cumulative distributions of KNN_distance and LOF features across training and testing sets. This further implicitly enables XGBoost to foresee the rapid changes in future modi operandi from the training set, which clearly is impossible in real-world applications, as reflected in the unrealistically high detection performance. The recall rates (86%–90%) and hence the F1 scores (88%–93%) all become high and stable regardless of changes in the training data size. We further repeated the experiments introduced at the beginning of Section V-A with a 60%/40% random partition of the training and testing sets, respectively, instead of a chronological split, also illustrating the results of random partitioning in Table 7. We observe that the presence of noisy features LOF and KNN_distance no longer deteriorates the fraud detection results of XGBoost. It can be observed that the F1 score is 86% for “XGBoost w/ **Other+Proposed**” and 80% for “XGBoost w/ **Other**,” which outperforms the counterpart experiments removing LOF and KNN_distance. Since changing modus operandi patterns should not be foreseen, it is inappropriate to assess a machine learning model with time-inhomogeneous data by random sampling or cross-validation.

We also applied the KS test to all **Other** and **Proposed** features as in Table 10 with the chronological data split of Table 7 to examine the similarity of these feature’s cumulative distributions in the training / testing datasets: the $p$ values of features other than LOF and KNN_distance are all high for both fraudulent and normal accounts. This implies that the distributions of these features in the training and testing sets are similar, and explains why the detection performance of XGBoost significantly improves when both LOF and KNN_distance are removed, as illustrated in Table 7. We examine the robustness of this finding by changing the proportion of the training set, as illustrated in Table 9 (blue cells). Excluding these two features stabilizes the recall rates and F1 scores, which are generally higher than the results in Table 9 (gray cells) regardless of changes in the training data size. We also use the discriminative learning studied in Bickel *et al.* [43] and Nair *et al.* [44] to show that removing noisy features makes the observations retrieved from training/testing sets more indistinguishable, as in Table 8;

**TABLE 9. Training XGBoost with various training set sizes. All numerical results are the averages from training and evaluating XGBoost five times. Gray and white cells denote data split chronologically and randomly, respectively, with all Other and Proposed features. Blue cells denote data split chronologically with KNN_distance and LOF excluded. Columns 6, 7, 8, and 9 list the *p* values generated by the KS test. The running times listed in the last column increase with the training set's size.**

| Training size | Accuracy | | | Precision | | | Recall | | | F1 score | | | Fraudulent KNN | | Fraudulent LOF | | Normal KNN | | Normal LOF | | running time (sec) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 50% | 99.68% | 99.91% | 99.76% | 100.00% | 89.68% | 72.36% | 20.48% | 86.75% | 66.27% | 34.00% | 88.15% | 69.18% | 7.91E-46 | 0.3757 | 1.88E-25 | 0.3957 | 1 | 1 | 0.3061 | 0.5789 | 18.62 | 39.92 |
| 60% | 99.60% | 99.92% | 99.78% | 100.00% | 92.61% | 74.19% | 1.49% | 86.67% | 68.66% | 2.94% | 89.49% | 71.31% | 9.61E-57 | 0.5467 | 1.81E-31 | 0.7165 | 1 | 1 | 0.992635613 | 0.4238 | 22.03 | 50.64 |
| 70% | 99.74% | 99.92% | 99.78% | 87.50% | 89.73% | 76.19% | 42.00% | 90.40% | 64.00% | 56.76% | 89.89% | 69.56% | 2.72E-33 | 0.5973 | 1.90E-16 | 0.6277 | 1 | 1 | 0.00359492 | 0.5415 | 47.42 | 58.05 |
| 80% | 99.81% | 99.92% | 99.81% | 90.91% | 90.04% | 90.90% | 58.82% | 90.30% | 58.82% | 71.43% | 90.08% | 71.43% | 1.07E-17 | 0.8889 | 1.03E-09 | 0.8779 | 1 | 1 | 0.490359292 | 0.4872 | 59.60 | 66.31 |
| 90% | 99.88% | 99.95% | 99.78% | 100.00% | 96.40% | 75.00% | 70.59% | 90.59% | 70.59% | 82.76% | 93.33% | 72.73% | 0.00331 | 0.6823 | 0.000551005 | 0.9276 | 1 | 1 | 0.027677963 | 0.7612 | 72.03 | 76.06 |

**TABLE 10. KS Test *p* values for each feature of (Non-)fraudulent accounts.**

| Feature | Fraudster $p$-value | Normal account $p$-value |
|---|---|---|
| LOF | 0 | 0.9926 |
| KNN_distance | 0 | 1 |
| Isolation_tree | 0.0165 | 0.1722 |
| Saving_gradient | 0.0464 | 0.1967 |
| Big_onetime_deal_count | 0.1812 | 1 |
| Total amount | 0.298 | 0.6482 |
| LT_signature | 0.3139 | 1 |
| Note_not_NE | 0.3288 | 0.8625 |
| ATM_transaction | 0.4053 | 0.7503 |
| Max_amount_same_day | 0.4318 | 0.582 |
| Debit_STDev | 0.4354 | 0.1647 |
| Amount_over_month | 0.4611 | 0.5644 |
| Average_daily_over_month | 0.4611 | 0.5644 |
| Average_over_2_months | 0.4611 | 0.4976 |
| Immediate_paid_out | 0.4875 | 0.9923 |
| Credit_STDev | 0.4972 | 0.4738 |
| Last_paid_out_larger_than_savings | 0.5245 | 1 |
| MAD | 0.6077 | 0.4055 |
| Weighted_fraud_coefficient | 0.6305 | 1 |
| Sensitive_single_amount_count | 0.6305 | 0.9676 |
| AVG_T_interval | 0.6782 | 0.5837 |
| Consecutive_transact_type_amount | 0.6824 | 0.2202 |
| Consecutive_transact_type_count | 0.6907 | 0.3842 |
| Amount_transaction_object_over_month | 0.7092 | 0.6002 |
| Amount_transaction_object_over_2_months | 0.7092 | 0.7444 |
| Sensitive_test_amount_count | 0.7438 | 0.4974 |
| Magic_number_check | 0.7852 | 0.9336 |
| Fits_Benford | 0.8079 | 1 |
| Max_number_same_day | 0.8116 | 0.9868 |
| 24hr_trade | 0.8861 | 0.7349 |
| Untrusted_frequent_trade_count | 0.9711 | 1 |
| Large_amount_count | 0.9951 | 0.5859 |
| MFTAC | 0.9983 | 0.3711 |
| LATM_signature | 0.9983 | 1 |
| Sensitive_daily_total_amount_count | 0.9999 | 1 |
| Zscore_outlier_count | 1 | 1 |
| Mahalanobis_anomaly | 1 | 0 |
| Bank_warned_branch | 1 | 1 |
| Zero_digit_freq | 1 | 1 |
| HFW_bank_Of_ATM | 1 | 1 |

that is, fraud detection models become less likely to learn time-varying patterns, which results in reduced predictability. Because the F1 score of the discriminative learning experiment is high when noisy features are not removed, the original account transaction data do exhibit a significant dataset shift problem. The F1 scores of the discriminative learning model decline after removing noisy features, which indicates that this problem has been alleviated. This explains why fraud detection results improve after removing noisy features.

To verify that handcrafted feature generation is essential when mass transaction data are not available under the constraint of privacy regulations, as in our case, we compare several feature synthesis algorithms proposed by [18], [21], [23], and [22], as illustrated in Table 11. Lucas *et al.* [18] generate features with multiperspective hidden Markov models to learn the monetary and recency properties of the transaction sequences from the credit card transaction histories. Kanter and Veeramachaneni [21] use a deep feature synthesis algorithm to generate features. Ebiaredoh-Mienye *et al.* [23] use the stacked sparse encoder to generate feature representation for each observation. We train the above three models with Bank L's transaction data to create features. These features are then used to train the classifier models listed in the second column of Table 11. Most automatic feature generation algorithms perform poorly, except that training the random forest with the deep feature synthesis algorithm yields good detection results. The performance of training gradient-boosting classification models with these algorithms is poorer than that of training with our proposed features. In addition, Esenogho *et al.* [22] use long-short term memory (LSTM) to capture temporal patterns from credit card transaction data. Then, they train an adaptive boosting model with these synthetic features to detect credit card fraud. However, the fraudulent detection performance in our experiment is also poor. This might be because most of these works study the detection of credit card fraud, which provides more aspects and large amounts of transaction information compared to the fund transfer data of banks' accounts detailed in Table 1. In addition, it might be a challenge for automatic feature generators to generate sophisticated behavior or segmentation features that require complex generation procedures and financial expertise in contrast to the simple RFMA features proposed in the past literature. Besides, these automatic feature generators typically require more computational resources, reflected in higher running times.

## B. ADDRESSING DATA IMBALANCE

To visualize the dataset imbalance[12] and the accounts' pattern features, each observation in our dataset is represented by a high-dimensional vector composed of the **Other** and **Proposed** features analyzed in Section IV except for KNN_distance and LOF. We used a 60%/40% chronological split for the training and testing datasets, respectively. We used principle component analysis (PCA) to project the observations represented by high-dimensional vectors to a two-dimensional plane, as illustrated in Figure 3. There are significantly fewer fraudulent observations (denoted by blue spots) than non-fraudulent ones (orange spots). Fraudulent observations are clustered in several subgroups, suggesting various modi operandi in the training dataset.

To mitigate learning bias toward the majority class due to data imbalance, we compared the resampling methods analyzed in [34], [35], and [27] with a Wasserstein GAN (see [24]). The ratio of fraudulent to non-fraudulent observations was rebalanced from 1 : 250 to 1 : 1 in the training data. Scatter plots for after-resampling training data are shown in Figure 3, and the corresponding fraud detection

[12] The ratio of fraudulent to non-fraudulent accounts in our dataset is 1 : 250.

**TABLE 11.** Comparisons with automatic feature engineering works.

| Compared result | Model | Accuracy | Precision | Recall | F1- score | running time (sec) |
|---|---|---|---|---|---|---|
| [18] | XGBoost | 99.44% | 35.63% | 46.96% | 40.52% | 1.58 |
| | Random Forest | 81.38% | 1.87% | 87.87% | 3.68% | 0.66 |
| | SVM | 90.60% | 3.37% | 80.30% | 6.47% | 34.46 |
| | LGBM | 99.57% | 46.66% | 31.81% | 37.83% | 2.08 |
| [21] | XGBoost | 99.59% | 33.33% | 1.49% | 2.85% | 379.79 |
| | Random Forest | 99.85% | 95.74% | 67.16% | 78.94% | 9.95 |
| | SVM | 18.85%* | 0.46% | 94.02% | 0.92% | 1597.57 |
| | LGBM | 99.59% | 50% | 1.49% | 2.89% | 21.75 |
| [23] | XGBoost | 77.81% | 0.09% | 48.19% | 0.19% | 16.64 |
| | Random Forest | 72.52% | 0.07% | 48.19% | 0.15% | 12.31 |
| | SVM | 4.59%* | 0.04% | 98.79% | 0.09% | 8187.70 |
| | LGBM | 85.93% | 0.08% | 44.57% | 0.28% | 25.42 |
| [22] | Adaboost | 90.86% | 50% | 1.85% | 3.57% | 1615.86 |
| Our result | Xgboost | 99.81% | 84.62% | 65.67% | 73.95% | 6.25 |
| | Random Forest | 94.76% | 6.20% | 85.07% | 11.56% | 0.64 |
| | SVM | 98.44% | 16.90% | 73.13% | 27.45% | 14.23 |
| | LGBM | 99.78% | 81.63% | 59.70% | 68.97% | 2.38 |

* Accuracy rates when training SVM using features generated by automatic feature engineering proposed in [21] and [23] are extremely low, whereas the accuracy rates for other experimental scenarios are generally high. To examine the correctness of these two experiments, we retrained these two models to confirm their convergence and then reran the prediction with the models on the training set data to find that the accuracy rates increased to 99.99% and 79.24%, respectively. Thus, our training process seems acceptable due to high accuracy rates when predicting the training data set. In addition, we changed the SVM settings in Table 5, for instance using a polynomial kernel with different degrees. However, despite the high accuracy rates on the training set, the rates are low on the testing set even after trying many different settings. To explore the reasons for this phenomenon, we found that the recall rates for these two scenarios exceed 90%, which is significantly larger than the recall rates for other scenarios. This indicates that the trained SVMs maintain high recall rates when identifying fraudulent accounts at the cost of mispredicting normal account samples. This explains the extremely low accuracy rates for predicting the testing data set since normal account samples account for about 99.6% of all accounts.

**TABLE 12.** Comparisons of resampling methods. The results were obtained by training XGBoost five times with Other and Proposed features excluding KNN_distance and LOF. The ratio of non-fraudulent to fraudulent accounts was adjusted to 1 : 1 by resampling. The first 60% (last 40%) of the data were split chronologically into training (testing) set data. The means of all standardized features' means and variances of the original data are the same as those of full oversampling. The running times are listed in the last column.

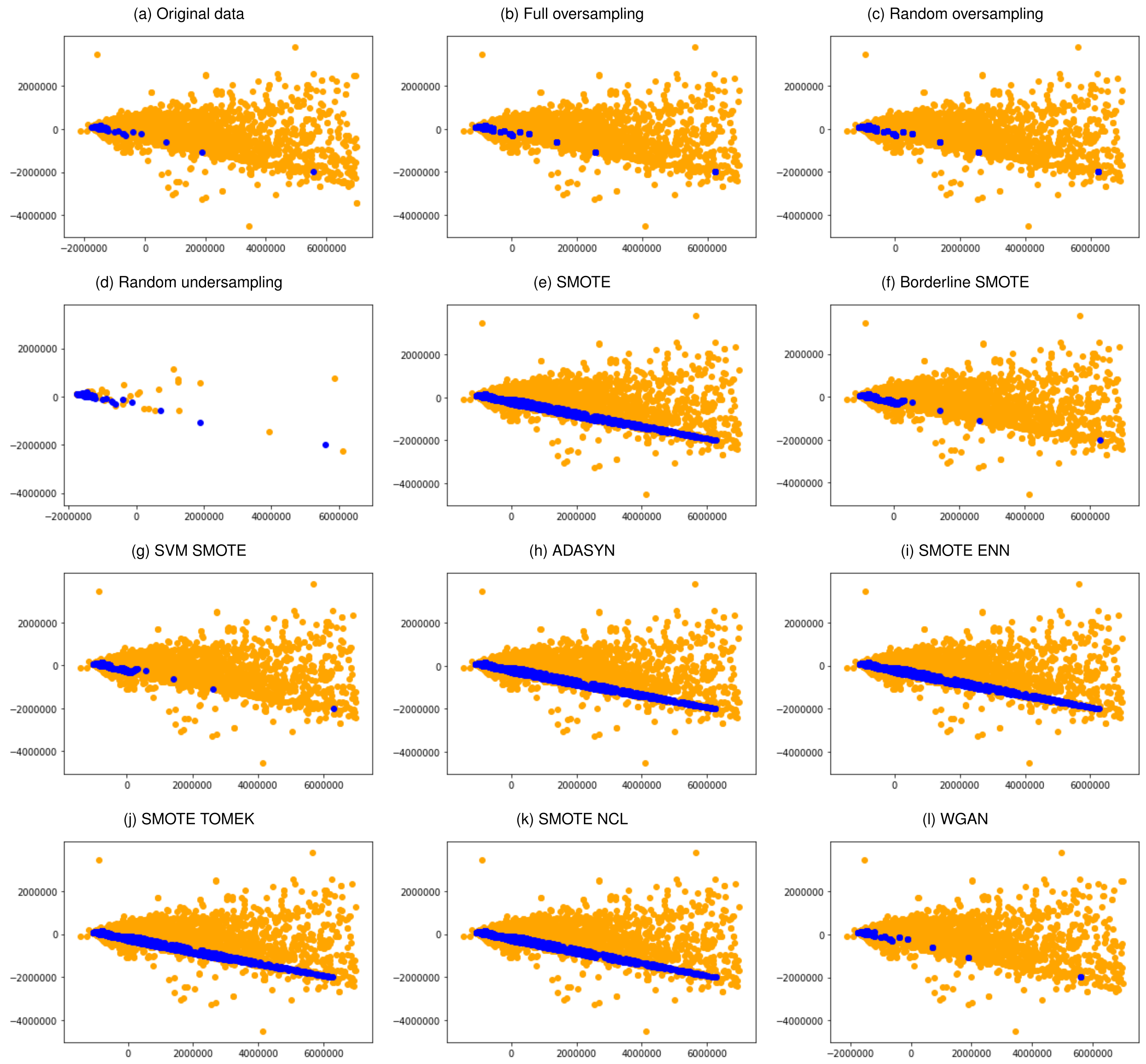
| **Resampling method** | **Accuracy** | **Precision** | **Recall** | **F1 score** | **AUC of precision-recall curve** | Mean of all standardized features' means | Mean of all standardized features' variances | running time (sec) |
|---|---|---|---|---|---|---|---|---|
| Full oversampling | 99.83% | 91.30% | 62.69% | 74.34% | 0.7689 | 0.566538223 | 0.573180329 | 50.6413 |
| Random oversampling | 99.82% | 91.11% | 62.12% | 73.87% | 0.7559 | 0.566604755 | 0.576216749 | 51.5147 |
| Random undersampling | 99.07% | 27.78% | 83.33% | 41.67% | 0.6394 | 0.566538223 | 0.573180329 | 0.3439 |
| SMOTE | 99.79% | 89.74% | 53.03% | 66.67% | 0.7357 | 0.414963722 | 0.393291514 | 52.0866 |
| Borderline_SMOTE | 99.78% | 85.71% | 54.55% | 66.67% | 0.7193 | 0.409573628 | 0.221649954 | 51.1346 |
| SVM_SMOTE | 99.79% | 88.10% | 56.06% | 68.52% | 0.7185 | 0.442363412 | 0.220753568 | 50.6327 |
| SMOTE_ENN | 99.79% | 88.10% | 56.06% | 68.52% | 0.7390 | 0.417634507 | 0.397403610 | 50.5292 |
| SMOTE_TOMEK | 99.81% | 92.50% | 56.06% | 69.81% | 0.7427 | 0.414384274 | 0.400725001 | 51.8532 |
| ADASYN | 99.78% | 89.47% | 51.52% | 65.38% | 0.7313 | 0.420792703 | 0.479966536 | 51.1461 |
| SMOTE_NCL | 99.81% | 90.48% | 57.58% | 70.37% | 0.7342 | 0.415822851 | 0.394679840 | 23.6258 |
| WGAN | 99.84% | 95.55% | 65.15% | 77.47% | 0.9696 | 0.552504013 | 0.545505743 | 42.9858 |

**TABLE 13.** Feature importance ranking in model "XGBoost with Other+Proposed without LOF & KNN_distance". Features with an importance of 0 are not listed here. The white cells correspond to features that belong to the proposed behavior or segmentation categories. The gray cells denote features belonging to the categories proposed in [7].

| **Features** | **Importance** | **Rank** |
|---|---|---|
| Branch_ID | 4.953581438 | 1 |
| LT_count | 2.084104460 | 2 |
| Most_frequent_transaction_type | 1.990122369 | 3 |
| Average_transaction_interval | 1.696345674 | 4 |
| Untrusted_frequent_trade_count | 1.401884328 | 5 |
| Second_most_frequent_transaction_type | 1.358480638 | 6 |
| Big_onetime_deal_count | 1.320201917 | 7 |
| Least_frequent_transaction_type | 1.292559594 | 8 |
| Max_number_same_day | 1.008210969 | 9 |
| Consecutive_transact_type_amount | 0.931961414 | 10 |
| Withdrawal_stdev | 0.893240978 | 11 |
| Max_amount_same_day | 0.796642561 | 12 |
| Saving_gradient | 0.765279028 | 13 |
| Isolation_tree | 0.723090867 | 14 |
| Suspicious_amount_count | 0.721782535 | 15 |
| 24hr_transaction | 0.717587597 | 16 |
| Note_not_empty_count | 0.717518352 | 17 |
| Fraud_factor | 0.693607278 | 18 |
| MAD | 0.625779145 | 19 |
| Deposit_stdev | 0.590518180 | 20 |

| **Features** | **Importance** | **Rank** |
|---|---|---|
| ATM_transaction | 0.547245151 | 21 |
| Amount_transaction_object_over_2_months | 0.452425771 | 22 |
| Amount_transaction_object_over_month | 0.429826250 | 23 |
| Immediate_paid_out | 0.422877468 | 24 |
| Sensitive_test_amount_count | 0.386304848 | 25 |
| Average_over_2_months | 0.384382749 | 26 |
| Last_paid_out_larger_than_savings | 0.372255388 | 27 |
| Consecutive_transact_type_count | 0.323095209 | 28 |
| Suspicious_ATM_bank | 0.255670551 | 29 |
| Total amount | 0.254789877 | 30 |
| LATM_count | 0.214525138 | 31 |
| Suspicious_branch | 0.209965529 | 32 |
| Sensitive_single_amount_count | 0.179052668 | 33 |
| Zero_digit_freq | 0.170525051 | 34 |
| Mahalanobis_anomaly | 0.136533557 | 35 |
| Amount_over_month | 0.129863498 | 36 |
| Sensitive_daily_total_amount_count | 0.117437642 | 37 |
| Fits_Benford | 0.088510716 | 38 |
| Average_daily_over_month | 0.053680240 | 39 |
| Large_amount_count | 0.033786134 | 40 |

performance is presented in Table 12. Full oversampling and random oversampling yield better F1 scores (74.34% and 73.87%) and higher areas under the curve (AUC) of the precision-recall curve (0.7689 and 0.7559) than other resampling methods since these two methods do not change the subgroup pattern of the modi operandi, as observed in



(a) Original data

(b) Full oversampling

(c) Random oversampling

(d) Random undersampling

(e) SMOTE

(f) Borderline SMOTE

(g) SVM SMOTE

(h) ADASYN

(i) SMOTE ENN

(j) SMOTE TOMEK

(k) SMOTE NCL

(l) WGAN

**FIGURE 3.** Scatter Plot of 2-Dimensional PCA projection of high-dimension resampled data. Orange and blue dots represent non-fraudulent and fraudulent observations, respectively. The name for each resampling method is listed in the lower part of each subfigure. The *x*- and *y*-axes represent the first and second PCA components, respectively.

Figures 3(b) and 3(c). Random undersampling produces the worst F1 score (41.67%) and AUC (0.6394), as it removes excessive non-fraudulent observations due to the extreme imbalance in the dataset, in which only 0.4% of accounts are fraudulent. Such removal harms the pattern learning of non-fraudulent accounts and reduces precision considerably. However, it also provides higher recall rates and uses less running time than other methods.

Unlike oversampling methods, which produce minority observations directly by replication, SMOTE-related methods produce observations by synthesizing new samples via linear interpolation. Our experiments in Table 12 suggest that these methods all yield lower F1 scores than the full / random oversampling methods. Figure 3 shows that SMOTE linear interpolation harms the learning of subgroup patterns of fraudulent accounts. Specifically, SMOTE, ADASYN, SMOTE_ENN, and SMOTE_TOMEK add synthesized fraudulent observations in places densely populated by non-fraudulent observations since these methods perform linear interpolation across all fraudulent observations. The resulting noise clearly reduces the ability to distinguish fraudulent accounts from normal ones. Borderline_SMOTE and SVM_SMOTE, however, perform linear interpolation selectively, and synthetic observations are added to the left part of the subfigure where fraudulent observations are dense, as shown in Figures 3(f) and 3(g). However, these two methods produce even lower F1 scores and AUCs.

The above problem might be explained by the argument in [45] that "*SMOTE does not change the expected value of the (SMOTE-augmented) minority class but it decreases its (minority class's) variability*" due to linear interpolation. Note that the decrease in the variability in fraudulent observations due to the use of SMOTE-related methods inhibits fraud detection models from identifying different fraud patterns. To estimate in greater detail how resampling methods influence the distributions of observations, we calculate the means of all standardized features' means and variances after applying different resampling procedures, as illustrated in columns 7 and 8 of Table 12. Applying full/random oversampling and random undersampling does not alter the means of features' means and variances, but applying SMOTE-related models could significantly reduce the mean of features' variances, especially for Borderline_SMOTE and SVM_SMOTE, because all synthetic observations are added to the tight area crowded with fraudulent accounts; that is, the left part of Figures 3(f) or 3(g). This phenomenon aggravates the effect of decreasing variance. Such unbalanced observation insertions (compared to other SMOTE-related methods) also decrease the mean of minority samples. The significant changes in the statistical properties could explain why these two SMOTE methods produce the poor detection results shown in Table 12. In conclusion, we suggest that SMOTE-related methods perform more poorly than simple oversampling methods if minor observations form several subgroups with severe data imbalance; this result is consistent with the findings in [46].

Furthermore, unlike the methods mentioned above, which use oversampling or linear interpolation to synthesize fraudulent observations, WGAN refines its synthetic samples via interaction between the generator and discriminator networks and does not insert improper observations, as illustrated in Fig. 3(l), nor does it decrease the means of features' means and variances, as in Table 12. Additionally, it somewhat prevents classification models from overfitting since synthetic observations are not exact replicas of the original observations. Thus, WGAN outperforms all other resampling methods in terms of the F1 score and AUC of the precision-recall curve, as illustrated in Table 12.

### C. INTERPRETABILITY AND FEATURE IMPORTANCE

The interpretability of fraudulent detection models is critical, as unreasonable false accusations and missed arrests could have serious legal ramifications. (Non-)fraudulent behaviors can also be profiled with interpretability to improve existing fraud detection rules and to sketch the cause-effect relationship of the model's fraud detection process. Currently, banks are required to abide by anti-money-laundering (AML) guidelines by screening transaction records according to specified rules. However, the rule-based model of our partner bank yields poor precision (40%) and recall (5.56%) rates. Training XGBoost with sophisticated handcrafted features based on the transaction patterns of (non-)fraudulent accounts and financial expertise can improve fraud detection results and capture cause-effect relationships to abide by AML guidelines. Feature importance reflects the strength of a relationship between a modus operandi and the feature. Table 13 measures and ranks the importance of the features defined in Table 2 (except for KNN_distance and LOF) by following the method proposed by Butaru *et al.* [28] published in a premium finance journal. All 10 features in our behavior and segmentation categories have nonzero importances, whereas 3 of the 30 RFMA features (see [8] and [7]) have an importance of 0. In addition, the two most important features belong to the behavior or segmentation categories. The experiments in Table 7 also suggest that training XGBoost with **Proposed** features outperforms XGBoost with **Other** features (with/without) KNN_distance and LOF. These results confirm the argument in [7], namely; namely, that good fraud detection can be achieved by careful feature engineering techniques even with simple classifier models.

## VI. CONCLUSION

Due to the limited amount of available transaction data and strong interpretability requirements, much of the literature addresses financial fraud detection by training a machine learning model with sophisticated handcrafted features instead of raw transaction data or automatic synthesized features. Handcrafted features generated in the literature can be divided into categories of recency, frequency, monetary, and anomaly (RFMA). This paper proposes behavior and segmentation-type features describing non-RFMA characteristics belonging solely to (non-)fraudulent accounts. Behavior-type features are generally constructed based on financial expertise, which can be interpreted as a knowledge base of an expert system. Segmentation-type features can be constructed based on statistical summaries of the classifications of raw transaction data, as in Tables 3 and 4, providing a good hint in future designs of automatic feature generation. We compare the performance to train popular classifiers, such as SVM, random forests, XGBoost, and LGBM, with features generated by automatic generation algorithms or proposed in the past fraud detection literature and in this paper to show the superiority of our proposed features. We analyze the features that cause XGBoost and LGBM to produce unstable detection results. These noisy features are time-inhomogeneous and are detectable using the Kolmogorov-Smirnov test. According to the experimental results, although SVM and random forest produce stable predictions without suffering from this unstable detection problem, XGBoost and LGBM yield better fraud detection results with fair interpretability by removing noisy features. In addition, the presence of noisy features reflects the time-inhomogeneous nature of the modi operandi. Improperly assessing the robustness of a machine learning model by generating training/testing sets with random sampling eliminates such time inhomogeneity and falsely produces good performance. To address data imbalance due to the small number of fraudulent accounts, we examine multiple resampling methods and WGAN. Because SMOTE-related methods apply improper linear interpolations on different modus operandi patterns, they decrease the variability of overall fraudulent observations and generate low-quality fraudulent observations. However, full (random) oversampling and WGAN avoid these problems and improve the detection results. The quality of our proposed features (categories) is verified by showing that the features in the proposed categories rank high according to the method proposed in [28] published in a premium finance journal.

## REFERENCES

[1] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognit.*, vol. 45, no. 1, pp. 521–530, Jan. 2012, doi: 10.1016/j.patcog.2011.06.019.

[2] C.-H. Tai and T.-J. Kan, "Identifying money laundering accounts," in *Proc. Int. Conf. Syst. Sci. Eng. (ICSSE)*, Jul. 2019, pp. 379–382.

[3] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Stat. Sci.*, vol. 17, no. 3, pp. 235–255, Aug. 2002.

[4] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams, "Transaction aggregation as a strategy for credit card fraud detection," *Data Mining Knowl. Discovery*, vol. 18, no. 1, pp. 30–55, Feb. 2009.

[5] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decis. Support Syst.*, vol. 50, no. 3, pp. 602–613, Feb. 2011.

[6] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature engineering strategies for credit card fraud detection," *Exp. Syst. Appl.*, vol. 51, pp. 134–142, Jun. 2016, doi: 10.1016/j.eswa.2015.12.030.

[7] B. Baesens, S. Höppner, and T. Verdonck, "Data engineering for fraud detection," *Decis. Support Syst.*, vol. 150, Nov. 2021, Art. no. 113492, doi: 10.1016/j.dss.2021.113492.

[8] X. Zhang, Y. Han, W. Xu, and Q. Wang, "HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture," *Inf. Sci.*, vol. 557, pp. 302–316, May 2021, doi: 10.1016/j.ins.2019.05.023.

[9] Y. Xie, G. Liu, R. Cao, Z. Li, C. Yan, and C. Jiang, "A feature extraction method for credit card fraud detection," in *Proc. 2nd Int. Conf. Intell. Auton. Syst. (ICoIAS)*, 2019, pp. 70–75, doi: 10.1109/ICoIAS.2019.00019.

[10] Y. Y. Hsin, T. S. Dai, Y. W. Ti, and M. C. Huang, "Interpretable electronic transfer fraud detection with expert feature constructions," in *Proc. CIKM Workshops*, 2021, pp. 1–11.

[11] K. Tripathi, "A review on knowledge-based expert system: Concept and architecture," in *Proc. IJCA Special Issue Artif. Intell. Techn.-Novel Approaches Practical Appl.*, no. 4, 2011, pp. 1–5.

[12] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," *J. Netw. Comput. Appl.*, vol. 68, pp. 90–113, Jun. 2016.

[13] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," *Exp. Syst. Appl.*, vol. 41, no. 10, pp. 4915–4928, Aug. 2014, doi: 10.1016/j.eswa.2014.02.026.

[14] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3784–3797, Aug. 2018, doi: 10.1109/TNNLS.2017.2736643.

[15] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Detecting credit card fraud using periodic features," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2015, pp. 208–213, doi: 10.1109/ICMLA.2015.28.

[16] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, "APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decis. Support Syst.*, vol. 75, pp. 38–48, Jul. 2015, doi: 10.1016/j.dss.2015.04.013.

[17] D. Cheng, S. Xiang, C. Shang, Y. Zhang, F. Yang, and L. Zhang, "Spatiotemporal attention-based neural network for credit card fraud detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 362–369.

[18] Y. Lucas, P.-E. Portier, L. Laporte, L. He-Guelton, O. Caelen, M. Granitzer, and S. Calabretto, "Towards automated feature engineering for credit card fraud detection using multi-perspective HMMs," *Future Gener. Comput. Syst.*, vol. 102, pp. 393–402, Jan. 2020, doi: 10.1016/j.future.2019.08.029.

[19] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, and O. Caelen, "Sequence classification for credit-card fraud detection," *Exp. Syst. Appl.*, vol. 100, pp. 234–245, Jun. 2018.

[20] A. Ivanov and G. Riccardi, "Kolmogorov-Smirnov test for feature selection in emotion recognition from speech," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 5125–5128, doi: 10.1109/ICASSP.2012.6289074.

[21] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *Proc. IEEE Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2015, pp. 1–10, doi: 10.1109/DSAA.2015.7344858.

[22] E. Esenogho, I. D. Mienye, T. G. Swart, K. Aruleba, and G. Obaido, "A neural network ensemble with feature engineering for improved credit card fraud detection," *IEEE Access*, vol. 10, pp. 16400–16407, 2022, doi: 10.1109/ACCESS.2022.3148298.

[23] S. A. Ebiaredoh-Mienye, E. Esenogho, and T. G. Swart, "Artificial neural network technique for improving prediction of credit card default: A stacked sparse autoencoder approach," *Int. J. Electr. Comput. Eng.*, vol. 11, no. 5, pp. 4392–4402, Oct. 2021.

[24] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, D. Precup and Y. W. Teh, Eds. 2017, pp. 214–223. [Online]. Available: http://proceedings.mlr.press/v70/arjovsky17a.html

[25] R. Moraffah, B. Moraffah, M. Karami, A. Raglin, and H. Liu, "Causal adversarial network for learning conditional and interventional distributions," 2020, *arXiv:2008.11376*.

[26] E. U. Savona and M. Riccardi, "Assessing the risk of money laundering: Research challenges and implications for practitioners," *Eur. J. Criminal Policy Res.*, vol. 25, no. 1, pp. 1–4, Mar. 2019, doi: 10.1007/s10610-019-09409-3.

[27] D. Vassallo, V. Vella, and J. Ellul, "Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies," *Social Netw. Comput. Sci.*, vol. 2, no. 3, May 2021, doi: 10.1007/s42979-021-00558-z.

[28] F. Butaru, Q. Chen, B. Clark, S. Das, A. W. Lo, and A. Siddique, "Risk and risk management in the credit card industry," *J. Banking Finance*, vol. 72, pp. 218–239, Nov. 2016, doi: 10.1016/j.jbankfin.2016.07.015.

[29] B. Wiese and C. Omlin, *Credit Card Transactions, Fraud Detection, and Machine Learning: Modelling Time With LSTM Recurrent Neural Networks*. Berlin, Germany: Springer, 2009, pp. 231–268.

[30] Y.-J. Zheng, X.-H. Zhou, W.-G. Sheng, Y. Xue, and S.-Y. Chen, "Generative adversarial network based telecom fraud detection at the receiving bank," *Neural Netw.*, vol. 102, pp. 78–86, Jun. 2018.

[31] S. Cao, X. Yang, C. Chen, J. Zhou, X. Li, and Y. Qi, "TitAnt: Online real-time transaction fraud detection in ant financial," *Proc. VLDB Endowment*, vol. 12, no. 12, pp. 2082–2093, Aug. 2019.

[32] D. Wang, Y. Qi, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, and S. Yang, "A semi-supervised graph attentive network for financial fraud detection," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2019, pp. 598–607.

[33] X. Li, S. Liu, Z. Li, X. Han, C. Shi, B. Hooi, H. Huang, and X. Cheng, "FlowScope: Spotting money laundering based on graphs," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, 2020, pp. 4731–4738, doi: 10.1609/aaai.v34i04.5906.

[34] R. Ghorbani and R. Ghousi, "Comparing different resampling methods in predicting students' performance using machine learning techniques," *IEEE Access*, vol. 8, pp. 67899–67911, 2020, doi: 10.1109/ACCESS.2020.2986809.

[35] N. F. Hordri, S. Sophiayati, N. Firdaus, and S. Mariyam, "Handling class imbalance in credit card fraud using resampling methods," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 11, pp. 1–7, 2018, doi: 10.14569/IJACSA.2018.091155.

[36] A. E. Khandani, A. J. Kim, and A. W. Lo, "Consumer credit-risk models via machine-learning algorithms," *J. Banking Finance*, vol. 34, no. 11, pp. 2767–2787, Nov. 2010, doi: 10.1016/j.jbankfin.2010.06.001.

[37] P. Addo, D. Guegan, and B. Hassani, "Credit risk analysis using machine and deep learning models," *Risks*, vol. 6, no. 2, p. 38, Apr. 2018, doi: 10.3390/risks6020038.

[38] Y. Zhang and P. Trubey, "Machine learning and sampling scheme: An empirical study of money laundering detection," *Comput. Econ.*, vol. 54, no. 3, pp. 1043–1063, Oct. 2019.

[39] Z. Chen, L. D. Van Khoa, E. N. Teoh, A. Nazir, E. K. Karuppiah, and K. S. Lam, "Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: A review," *Knowl. Inf. Syst.*, vol. 57, no. 2, pp. 245–285, Nov. 2018, doi: 10.1007/s10115-017-1144-z.

[40] K. Chitra and B. Subashini, "Data mining techniques and its applications in banking sector," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 3, no. 8, pp. 219–226, Aug. 2013.

[41] R. Liu, X.-L. Qian, S. Mao, and S.-Z. Zhu, "Research on anti-money laundering based on core decision tree algorithm," in *Proc. Chin. Control Decis. Conf. (CCDC)*, May 2011, pp. 4322–4325, doi: 10.1109/CCDC.2011.5968986.

[42] J. Frery, A. Habrard, M. Sebban, O. Caelen, and L. He-Guelton, "Efficient top rank optimization with gradient boosting for supervised anomaly detection," in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Computer Science), vol. 10534, M. Ceci, J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, Eds. Cham, Switzerland: Springer, 2017.

[43] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning for differing training and test distributions," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 81–88.

[44] N. G. Nair, P. Satpathy, and J. Christopher, "Covariate shift: A review and analysis on classifiers," in *Proc. Global Conf. Advancement Technol. (GCAT)*, Oct. 2019, pp. 1–6.

[45] R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data," *BMC Bioinf.*, vol. 14, no. 1, Dec. 2013, doi: 10.1186/1471-2105-14-106.

[46] S. Tyagi and S. Mittal, "Sampling approaches for imbalanced data classification problem in machine learning," in *Proc. ICRIC*, 2020, pp. 209–221, doi: 10.1007/978-3-030-29407-6_17.

**YU-YEN HSIN** received the M.S. degree from the National Yang Ming Chiao Tung University, in 2021. His research interest includes machine learning in finance.

**TIAN-SHYR DAI** received the Ph.D. degree from the Department of Computer Science, National Taiwan University. He was the Chairperson of the Department of Information Management and Finance, from 2016 to 2019, and the Director of the Taiwan Association of Business Schools, from 2018 to 2020. He has been a Faculty Member of Beta Gamma Sigma, since 2021. He is currently a Full Professor at the Department of Information Management and Finance, NYCU. He is also a Research Member of the Risk and Insurance Research Center, NCCU. He is a Senior Fellow of Advance HE. His research interests include financial engineering and financial technology.

**YEN-WU TI** received the B.E. degree in mathematics from Tamkang University, in 1995, the M.E. degree in applied mathematics from the National Chiao Tung University, Hsinchu, Taiwan, in 1997, and the Ph.D. degree in computer science and information engineering from the National Taiwan University, Taipei, Taiwan, in 2009. He is currently an Associate Professor at the College of Artificial Intelligence, Yango University, China. His research interests include machine learning and algorithms.

**MING-CHUAN HUANG** received the B.E. degree in computer science from the National Taiwan Normal University, Taipei, Taiwan, in 2020. She is currently a Graduate Student at the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. Her research interest includes anti-money laundering.

**TING-HUI CHIANG** received the Ph.D. degree in computer science from the National Yang Ming Chiao Tung University, Taiwan, in 2018. He has been an Assistant Professor at the Department of Information Engineering and Computer Science, Feng Chia University, Taiwan, since 2019. His research interests include artificial intelligence and indoor localization. In AI research, he focuses on activity recognition, video inpainting, and fintech services. For localization research, he focuses on wireless localization, pedestrian dead reckoning, acoustic localization, particle filters, and AI-based localization models.

**LIANG-CHIH LIU** received the Ph.D. degree in finance from the National Chiao Tung University, in 2016. He joined the Department of Information and Finance Management, National Taipei University of Technology, in 2017, as an Assistant Professor. His research interests include computational finance, credit risk issues associated with corporate debt structure, issues concerning optimal call policy of corporate bonds, and text mining.