Machine Learning (2024) 113:5087–5108
https://doi.org/10.1007/s10994-023-06354-5

Fraud detection with natural language processing

Petros Boulieris[1] · John Pavlopoulos[1,2] · Alexandros Xenos[1] · Vasilis Vassalos[1]

Received: 7 June 2022 / Revised: 30 April 2023 / Accepted: 16 May 2023 /
Published online: 19 July 2023
© The Author(s) 2023

Abstract
Automated fraud detection can assist organisations to safeguard user accounts, a task that is very challenging due to the great sparsity of known fraud transactions. Many approaches in the literature focus on credit card fraud and ignore the growing field of online banking. However, there is a lack of publicly available data for both. The lack of publicly available data hinders the progress of the field and limits the investigation of potential solutions. With this work, we: (a) introduce *FraudNLP*, the first anonymised, publicly available dataset for online fraud detection, (b) benchmark machine and deep learning methods with multiple evaluation measures, (c) argue that online actions do follow rules similar to natural language and hence can be approached successfully by natural language processing methods.

Keywords Fraud detection · Natural language processing · E-banking · Feature engineering · Varying class imbalance

1 Introduction

Fraud detection systems evaluate each transaction in order to identify fraudulent ones. A bank employee can then assess the risk in accordance with their safety threshold and choose a course of action (e.g. block or ask for additional information). With Internet and mobile banking achieving widespread use and reaching several million transactions

Editors: Nuno Moniz, Nathalie Japkowicz, Michal Wozniak, Shuo Wang.

✉ Petros Boulieris
petrosboulieris@gmail.com

John Pavlopoulos
ioannis@dsv.su.se

Alexandros Xenos
a.xenos20@aueb.gr

Vasilis Vassalos
vassalos@aueb.gr

[1] Department of Informatics, Athens University of Economics and Business, Athens, Greece

[2] Department of Computer and Systems Sciences, Stockholm University, Stockholm, Sweden

per day, it is impossible for a human to monitor and detect all cases. However, preventing and detecting fraud in a timely manner is crucial to build and safeguard customer confidence in those platforms, as well as to the bank in order to avoid chargebacks. Hence, automated fraud detection can detect potentially illegitimate transactions and become the last line of defense before an employee has to step in and rectify the situation.

Traditional approaches to fraud detection are mostly rule-based (Panigrahi et al. 2009; Fawcett and Provost 1997) (binary features that immediately flag a transaction e.g. IP location over X km away from the user's registered address), which, while simple to implement, have intrinsic limitations. For instance, they cannot adapt to evolving fraud patterns without human intervention and require domain experts to engineer and update features. Fraud detection solutions that are based on machine (Lucas 2019; Patel et al. 2019; Wang et al. 2017; Mehana 2020) and deep (Carminati et al. 2018; Achituve et al. 2019) learning do not suffer from these drawbacks, however, they have only been investigated on top of features that require heavy engineering to be extracted. For example, in Baesens et al. (2021), the authors fit a von Mises distribution on the timestamps of each user's transactions to construct confidence intervals for the time periods that transactions generally take place for each user.

In this paper we tackle fraud detection by focusing on online and mobile banking transactions. Following the mainstream suggested in related work, we initially approached the task as imbalanced binary classification. Transactions were modelled as stand-alone events, ignoring any user actions that precede them. Also, we undertook the time-consuming, sophisticated, feature engineering that is required to extract commonly adopted user-profiling features (Fawcett and Provost 1997). Compared to this baseline, we extracted the online API calls (e.g., `'/login/'`, `'/logout/'`) made by each user and recorded on the Bank's server. For each user session that involved a transaction, we concatenated all the API calls into one action sequence that ended with the transaction (i.e., transfer or payment), and we labelled each sequence as fraudulent or not according to the Bank's decision regarding the respective ending transaction. Here, we consider the chain of user actions (e.g., login, logout, balance check, recent history) that lead to a transaction of some type (e.g., generic transfer, rent or bill payment, instant payments) and use the entire sequence (alongside other features) to evaluate the transaction. Extracting sequences in this way allowed us to: (a) cast fraud detection as a sequence classification problem that required minimum feature engineering, and (b) be the first to share our dataset for public use.

Fraud detection is applicable in two settings, online and offline. The former regards real-time detection, which is expected to function proactively, alarming an employee to step in. This setting is better addressed by high-precision algorithms. The latter concerns the evaluation of historical data, in order to detect possibly missed frauds. This setting is better addressed by high-recall methods. In contrast to most published related work that disregards this observation, we evaluate all our methods in both settings. Experimentation on our dataset demonstrates that online actions present similarities with natural language and that NLP-based features can leverage the performance of fraud classifiers, outperforming existing methods while requiring minimal engineering and respecting privacy more.

Overall, our contributions are the following:

- We introduce *FraudNLP*, the first anonymised publicly available dataset for fraud detection, which is based on actions of users preceding a transaction, consisting of 105,303 transactions, generated by 2,000 users.

- We benchmark machine and deep learning algorithms on our dataset, by also assessing online and offline detection, which are important but have been overlooked by prior studies.
- We show that the use of privacy-safe NLP-based features leverages the performance of machine learning and can outperform the state-of-the-art.
- By assessing our best-performing classifier on different class imbalance settings, we show that the problem is much more challenging for higher imbalance, closer to the true nature of the task.

The remainder of the article first discusses related work and then presents the new dataset. Following, an empirical analysis and a discussion section are provided. A summary of our findings concludes this study.

2 Related work

The vast majority of the fraud detection studies suggested in the literature disregard data from e-banking and mainly focus on credit card fraud. Below we first present fraud detection studies with data coming from online actions, which is most related to our work, and then from other types.

2.1 Online fraud

Michele Carminati et al. (2015) propose BankSealer, a system that uses a semi-supervised approach to rank user transactions by suspiciousness. They mainly utilize anomaly detection methods to build user-specific behavioral profiles with respect to their transaction history, without using any sequential information. In another paper, Michele Carminati et al. (2018) propose a framework called FraudBuster for detecting financial frauds that involve stealing small amounts of funds over time. Their framework models the user's spending pattern over time and detects frauds as transactions that deviate from the learned model and change the user's spending profile.

Kovach and Ruggiero (2011) propose a system that creates a risk score by combining changes in behavior at the local (user) level and at the global level, among all users in the bank. They, too, treat transactions as points in time with no action history leading up to them and introduce contextual information by means of heavily engineered statistical features (differential analysis to quantify abnormality on the local level, probabilistic model on the global level, and Dempster-Shafer theory to combine the two). Additionally, they require users to download separately an application to facilitate device fingerprinting, which makes this method more difficult to implement, especially if one wants to somewhat respect privacy.

Deviating from prior work in the field that focuses on transaction sequences (Wang 2021; Forough and Momtazi 2022, 2021), we leverage the sequence of user actions (that precede a transaction), in order to evaluate the legitimacy of a transaction. This formulation allowed us to cast fraud detection as a sequence classification problem, which required limited feature engineering. Our results show that the proposed feature engineering leads to well-performing solutions, and even *surpasses the state of the art* when it is combined with

simple anomaly detection features. Our features are anonymous by nature, hence with this work *we release the first publicly available dataset for online fraud detection*.

2.2 Other fraud sources

Related to our work are also other types of fraud, such as credit card fraud. Most of the available published work regarding its detection uses transaction logs as input (Patel et al. 2019; Forough and Momtazi 2021; Nguyen et al. 2020; Zhang et al. 2020; Rinku et al. 2023; Zamini and Montazer 2018), which require heavy data engineering. The benefits of employing data engineering were showcased in Baesens et al. (2021), where the authors used the Recency, Frequency, Monetary (RFM) principle to extract features and to effectively capture the spending behavior per customer. Their results showed a significant improvement in classification performance, even for simpler classifiers like logistic regression. The authors also demonstrated that the addition of features based on anomaly detection techniques yielded improved results. We also employ anomaly features and we outperform their approach in the same evaluation measure and imbalance settings.

Recently, the task of fraud detection was approached by Recurrent Neural Networks which could extract information from the history of card transactions for each user (Branco et al. 2020; Roy et al. 2018). Achituve et al. (2019) treat the history of recent transactions of each user as a sequence. By using attention-based RNNs they get increased performance and attention scores, which they use to provide interpretability to the response of their classifier. They encode the metadata of each transaction (day of week, hour of day, amount, device identifier, etc.) to several variables and then begin to learn embedding vectors. The transactions were batched in sequences, enabling historical information of variable width to be used for each client. The results of Jurgovsky et al. (2018) show that LSTM improves detection accuracy in offline transactions when compared to their baseline random forest classifier and that combining sequential and non-sequential learning methods could lead to more effective detection of fraud. (Kunlin 2018) proposed a novel fraud detection algorithm called FraudMemory, which used state-of-art feature representation methods to better depict users and logs with multiple types in financial systems. The model innovatively captures the sequential patterns of each transaction and leverages memory networks to improve performance. The incorporation of memory components in FraudMemory enhanced its adaptability to concept drift in evolving environments.

In the field of sequence classification, a credit card fraud detection model was proposed by Forough and Momtazi (2022) using deep neural networks and probabilistic graphical models. The study compared their model with the baseline using real-world datasets and found that considering hidden sequential dependencies among transactions and predicted labels improved results. A novel undersampling algorithm was also introduced and demonstrated promising results compared to other oversampling and undersampling methods. Similar to this, and on the topic of behavioral modelling for fraud detection, Wang (2021); Rodríguez et al. (2022) both come very close to us with respect to their philosophy and feature set. Their work shows that unauthorized behavior is not always necessary for fraud detection in online payment services. Their proposed solution is an account risk prediction scheme that tries to predict fraud before it occurs by analyzing a user's historical transaction sequence.

In e-commerce fraud detection, although not in a banking context, a similar line of work has been suggested by effectively treating user actions as events in time (Wang et al. 2017). We also exploit the temporal nature of user actions, but we approach each transaction as *a*

chain of smaller events leading to a payment, as opposed to a monolithic entity that contains some metadata (e.g., RFM information). This approach allowed us to cast the problem of fraud detection as a sequence classification task.

3 The new dataset

The *FraudNLP* dataset presented with this study[1] regards physical persons who interacted with a bank in Europe through its internet and mobile banking platforms.[2] Below, we discuss first the dataset development process. Then, we discuss the numerical features that are often extracted in related work (Baesens et al. 2021; Wedge et al. 2019) and features related to anomaly detection that have been shown to improve the detection performance (Baesens et al. 2021). The sequential data introduced with this work are presented last.

3.1 The dataset development process

We examined transactions within the time frame between 1st of February and 31st of October 2020, which includes all recorded cases of fraud, along with many millions of legitimate transactions. Out of all the transactions that were recorded during this period, 101 were certified by a bank employee as fraudulent. First, we selected 10,000 users that had no fraudulent transactions during these nine months, chosen at random within the bank servers. Many users had not interacted with the bank services enough to provide us with meaningful information on their spending habits, so their logs had to be removed. A cutoff of at least 12 transactions was used to filter them, with the final dataset containing logs from 2,000 users. The logs of 97 users were then added, who were users that had at least one transaction certified as fraudulent by a bank employee within this time frame. We note that this process leads to an underestimation of the true fraction of the users with fraudulent transactions because the 2,000 users that we used are only a small sample of the users with no registered fraudulent activity within that time frame. The same scarcity applies at the level of fraudulent transactions, where only 101 instances of fraud exist within the 105,303 transactions in total (0.096%). We also note, however, that not all fraudulent transactions are captured by the bank and, while the confirmed fraud cases that we have can be considered perfectly accurate, unlabeled and unreported cases may be present in this dataset.

Due to the sensitive nature of the raw dataset, all sensitive information was stripped, including usernames, IBAN numbers, names, and credit card information, and we had to work with the fully anonymised version. This constraint was not present in the studied literature since fully personalized features that depend on the client (e.g. graph-based approaches, account balance, detailed spending information, credit score) and spatiotemporal information (e.g. timestamps, number of transactions in a given time frame, GPS data of both transaction parties) as well as device characteristics were used. Comparatively, we had access to a very limited subset of that information, which significantly increases the difficulty of capturing illegitimate transactions. To make some use of the personal information before deleting it, several features were engineered. These can be considered behavioral features since they provide an approximation of a user's typical spending habits but

[1] Github repository: https://github.com/pboulieris/FraudNLP.

[2] The name of the bank is kept confidential to avoid compromising the identities of the clients.

Table 1 Artificial examples of user actions, organised as sequences, that have been labelled as fraudulent or not. The transaction amount (TA) in log(EUR), the elapsed time of the transaction (TT) in ms, and the time elapsed between the actions (TBA) in ms are also shown

Actions	TBA	TT	TA	Fraud
`['/login', ..., / transfer']`	[0, ..., 10]	200	80.0	Yes
`['/login', ..., / loans/pay']`	[0, ..., 140]	202	120.5	No

do not expose his identity directly, therefore they respect our privacy constraints. As will be discussed next, the raw usage logs on the bank servers were then processed to form sequences of actions leading to transactions (see Table 1).

3.1.1 Recency, frequency, monetary

The most common features extracted in fraud detection are numerical features, modeling the user's behavior based on the Recency, Frequency, Monetary (RFM) principle, as in Baesens et al. (2021). Following their work, we implement features that adhere to it. Starting from the oldest available transaction per user (i.e., 9 to 11 months in this dataset), an expanding frequency table was created. This table, after a burn-in period of one month, captures many of the persons and entities each client interacts with normally, including one-time and recurring payments like rent, subscriptions, or mortgage payments. Furthermore, the logarithm of each transaction amount (in EUR) was extracted based on the client's request and the respective server response. These were standardised using the mean and standard deviation of the amounts in the training set. Requests and responses also contained beneficiary information, which was used to create the expanding relative frequency tables for each beneficiary.

The same methodology was used to process device characteristics and operating systems, IP address, and application type (iOS or Android).

3.1.2 Unsupervised anomaly detection

Unsupervised anomaly detection features can increase performance in detecting fraudulent transactions, through distance metrics (e.g. Z-Scores, Mahalanobis distance), statistical laws, and anomaly scores (e.g., k-NN, LOF) (Baesens et al. 2021). In this work, we opted for the Isolation Forest algorithm (Liu et al. 2009),[3] which works by randomly selecting a feature, then randomly selecting a split value between the maximum and minimum values of the selected feature. This recursive partitioning can be represented by a tree structure, and the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node. This path length, when averaged over a forest of trees, can be considered a measure of normality, and is also the output of the algorithm.

[3] We could not reproduce exactly the same features that were presented in Baesens et al. (2021), due to issues with accessing sensitive information at a later stage of the project.

Table 2 Features based on action sequences (top), RFM (middle) and anomaly detection

Feature	Type	Dimensions
User action sequence	Integers	256
Time between actions (ms)	Integers	256
Log of transaction amount	Float	1
Time to execute transaction (ms)	Float	1
Device frequency	Float	1
IP address frequency	Float	1
Application frequency	Float	1
Beneficiary frequency	Float	1
Isolation forest score	Float	1

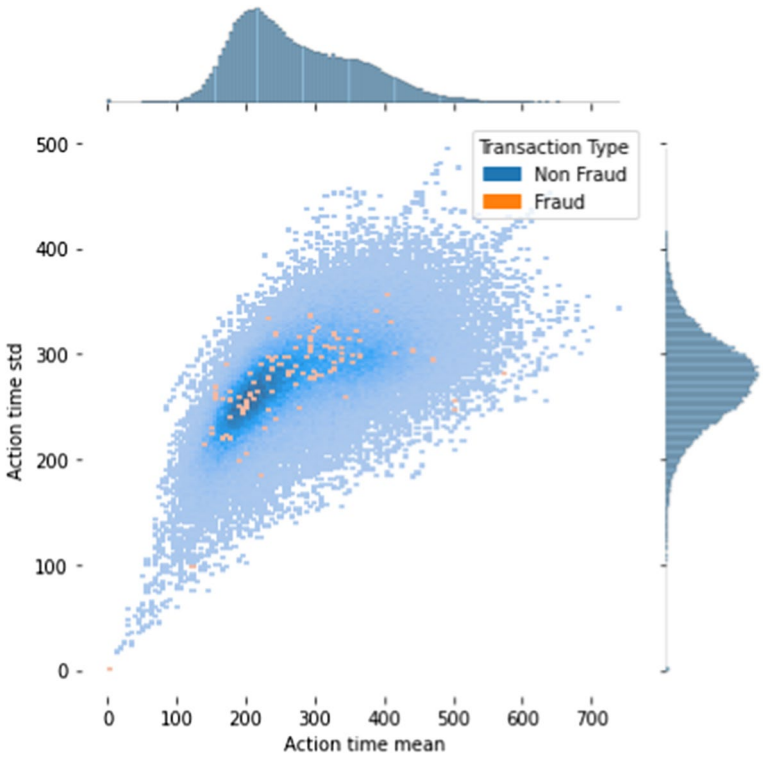

Fig. 1 Histograms of the mean (X) and st. deviation (Y) of the time of actions (in seconds), for fraudulent (in orange) and non-fraudulent (in blue) transactions

A forest of 100 trees was fitted on the training subset, and its predicted anomaly scores were extracted for each training observation.

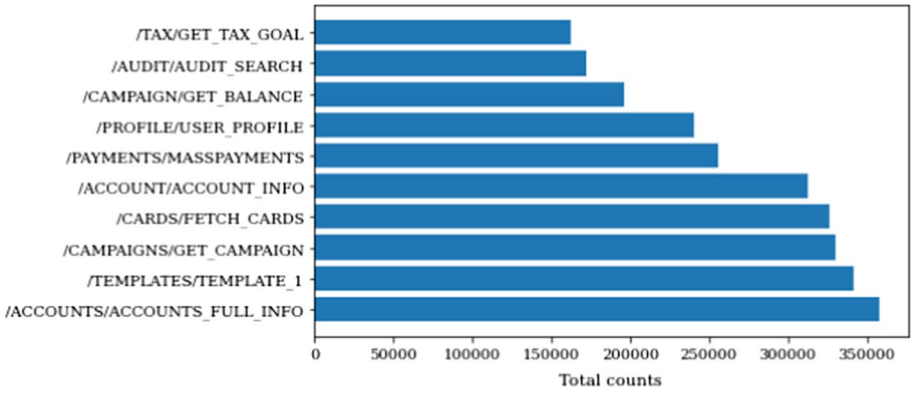

Fig. 2 The most frequent actions in the dataset

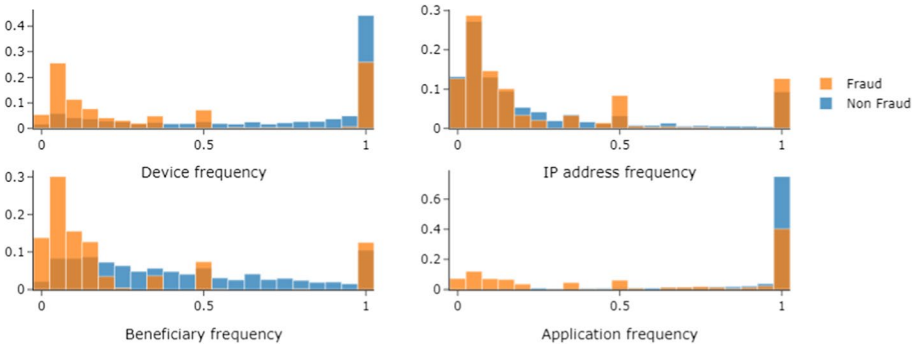

Fig. 3 Histograms of the numerical feature frequency (X-axis) for fraudulent (orange) and non-fraudulent (blue) transactions (Color figure online)

3.1.3 From user actions to action sequences

Each row in the raw log files comprises information regarding the interaction between a user and the bank servers, for example about one-time and recurring payments (e.g., rent or mortgage payments). This information regards API calls, timestamps, the application type, device characteristics, operating system specifications, IP address, client requests, and server responses, etc. Following the data engineering carried out by Wang (2021), rows were grouped by user ID, sorted by their timestamp, and segmented based on the transactions; i.e., a sequence of user actions ends with a transaction in EUR. For some classifiers (e.g. LSTM) each sequence was truncated or padded to a length of 256 actions for training. If the sequence of actions was considered fraudulent activity by the Bank, we label this sequence as fraudulent, otherwise is non-fraudulent. This processing essentially allows us to cast the original fraud detection problem into a sequence classification task. The overall list of features that were created is summarized in Table 2.[4]

[4] We note that the time between actions, although potentially useful, did not improve performance in preliminary experiments and was not investigated further in this work.

Table 3 The class balance in the training, validation and test subsets of our dataset, presenting the absolute number (and the percentage) per class. Fraudulent instances are less than 0.1% in all subsets

	Transaction label	
	Normal (%)	Fraudulent (%)
Training	63,178 (99.903)	61 (0.097)
Validation	21,060 (99.905)	20 (0.095)
Test	21,059 (99.905)	20 (0.095)

3.2 Exploratory analysis

As can be seen in Fig. 1, the fraudulent action sequences (in orange) are few and scattered among the (blue) non-fraudulent ones, with regard to the time elapsed from action to action within a single sequence. When focusing on the actions inside the sequences, we find that the five most frequent actions (Fig. 2, lowermost) are also the most frequent per class. One exception to this observation is '/LOANS/LIST', which is the 6th most frequent in fraudulent sequences while it is the 7th most frequent for the many more non-fraudulent ones. Also, we note that '/CARD/FETCHCARDS' is much more frequent in fraudulent transactions (10th), compared to non-fraudulent ones (21st).

Another interesting statistic is the distribution of each numeric feature by transaction status (Fig. 3). It becomes evident that our features uncover the different behavior that we expected fraudulent transactions to express. In the device frequency histogram, we note two distinct areas where fraud occurs, the very high and very low-frequency zones. The former can be attributed to device theft where the same device is used to commit fraud, while the latter can be attributed to account takeover, where a new device is used. On the other hand, in the IP address frequency histogram, there exists little difference in the distribution of legitimate and illicit transactions except for the medium and high-frequency areas. It is likely that this is because both in account takeover and in device theft scenarios, a new IP address is used, and thus we do not see the two distinct areas we see in the other features.

Although the statistics of the sequences themselves did not reveal many interesting patterns, more discrepancies are expected in action sub-sequences, given that fraudsters may be using the same actions (i.e., not to draw attention), but in a different order (e.g., to view the available cards). When exploring the frequency of action trigrams (i.e., sequences of three consecutive actions), we report that the most frequent instance is often different between the two classes, when we take equally sized non-fraudulent samples. By repeating this sampling one thousand times, we report that this is a statistically significant finding (P-value = 0.02).

4 Empirical analysis

In all our experiments we used a train/development/test stratified split of 60/20/20 percent, respectively, and performed Monte Carlo 5-fold Cross-Validation. We trained and assessed four machine learning classifiers on our data, in order to predict if a transaction is fraudulent or not. We used Logistic Regression (LR), Random Forests (RF), k nearest neighbors (kNN), and Support Vector Machines (SVM).

4.1 Evaluation measures

The high class-imbalance (Table 3) increases the challenge of the task. For assessment, we opted for evaluation metrics that are not affected by the skewed nature of the problem. Namely, we used the F_1 score and Area Under the Precision-Recall Curve (AUPRC). We chose the latter over the area under the ROC curves (ROC-AUC) (Saito and Rehmsmeier 2015) due to its relative insensitivity to class balance. A majority classifier in this task would have a precision of 0.096%, which would accurately be reflected in the AUPRC (0.096%), as opposed to the ROC-AUC (50%).

Both F_1 and AUPRC are employed in related studies (Baesens et al. 2021), but they do not differentiate between the high-Precision and the high-Recall zone. This is problematic because fraud is most often detected either in real-time (online) when Precision matters most, or offline when it is Recall that matters most. Hence, we suggest that methods are evaluated with the high-precision F_{05} for the online and with the high-recall F_2 for the offline setting. For all three F-scores, and for each fold, we fine-tuned the classification threshold on the development set.

4.2 Experimental results

First, we experimented with RFM-based features, then with a temporal (NLP-based) formulation. The suggested temporal data formulation, allowed us to leverage also deep learning (neural) approaches. In a final experiment, we combined RFM with NLP-based features.

4.3 Machine learning with RFM-based features

The results of our experiments using machine learning on top of RFM-based features (see Sect. 3.1) are shown in Table 4. RF outperforms the other models, achieving 25.1% AUPRC. LR is the second best model in this setting with 17.2%, while kNN and SVM perform poorly. RF performs equally well in the online (32.1%) and offline (34.2%) setting, but lower in $F1$ (28.5%).

4.4 Machine learning with NLP-based features

The most frequent user action n-grams differ between legitimate and fraudulent instances, and the difference is statistically significant (see Sect. 3.2). Hence, we hypothesised that term frequency-inverse document frequency (TF-IDF) features, based on unigrams, bigrams, and trigrams, can potentially provide the same four machine learning algorithms with adequate input information in order to classify a sequence of actions as fraudulent or not. We generated a vocabulary of 21724 unique n-grams by training on the entire training set. Table 5 presents the experimental results of this hypothesis.

The $F1$ score of the best-performing RF improved (+ 0.15) when we switched from RFM to NLP-based features. Improvement can also be observed in the online evaluation, but not in AUPRC or the offline setting. SVM with the NLP-based TF-IDF features improved significantly, even outperforming the traditional, RFM-based RF in all measures. The other two algorithms also registered improvements in all measures. This is particularly

Table 4 Assessment machine learning fraud classifiers with standard (AUPRC, F_1), online (F_{05}) and offline (F_2) measures (best in bold) using RFM-based features (F:RFM)

F:RFM	F:NLP	Method	AUPRC	$F_{0.5}$	F_1	F_2
☑	☐	LR	0.172	0.192	0.226	0.291
☑	☐	RF	**0.251**	**0.321**	**0.285**	**0.342**
☑	☐	*k*NN	0.040	0.155	0.176	0.204
☑	☐	SVM	0.001	0.001	0.002	0.005

Bold values indicate the highest value of each column

Table 5 Assessment of machine learning fraud classifiers with standard (AUPRC, F_1), online (F_{05}) and offline (F_2) measures (best in bold) using NLP-based n-gram features (F:NLP)

F:RFM	F:NLP	Method	AUPRC	$F_{0.5}$	F_1	F_2
□	TF-IDF	LR	0.222	0.293	0.305	0.326
□	TF-IDF	RF	0.211	0.333	0.300	0.269
□	TF-IDF	*k*NN	0.056	0.141	0.138	0.162
□	TF-IDF	SVM	**0.331**	**0.467**	**0.411**	**0.438**

Bold values indicate the highest value of each column

Table 6 Assessment of machine learning fraud classifiers with standard (AUPRC, F_1), online (F_{05}) and offline (F_2) measures (best in bold) using centroids of NLP-based FastText action embeddings

F:RFM	F:NLP	Method	AUPRC	$F_{0.5}$	F_1	F_2
□	FastText Emb. Centroids	LR	0.068	**0.132**	**0.133**	**0.163**
□	FastText Emb. Centroids	RF	0.039	0.097	0.084	0.149
□	FastText Emb. Centroids	KNN	0.013	0.083	0.096	0.113
□	FastText Emb. Centroids	SVM	**0.119**	0.093	0.078	0.068

Bold values indicate the highest value of each column

interesting in the light that the employed features in this case only encode the communication between the (user's) front and (Bank's) back end, without any information regarding the user's ID and historical records.

Replacing the TF-IDF representation with other popular NLP techniques, such as (sub) word embeddings, did not yield any improvements. In specific, we trained a FastText model (Bojanowski et al. 2017) on our transactional "corpus", assuming that each action sequence is the corresponding of a tokenised text. Each action was then mapped to a dense embedding and for each sequence, we could compute the centroid as a representation of the respective instance. Table 6 shows the results. Note that the NN architectures that will be presented later trained their own dense representations using a Keras Embedding layer.

A comparison of Tables 5 and 6 reveals that the results of machine learning algorithms that operate on top of centroids of action dense embeddings are not promising. However, neural algorithms are often employed in NLP tasks, outperforming their machine-learning counterparts when operating on top of dense representations. To investigate this assumption, we experimented first with a stacked Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) with a feed-forward neural network (FFNN) on top.[5] Secondly, we used a Convolutional Neural Network (CNN) (LeCun and Bengio 1995), which applies a 1D convolution on the input sequence. We used 16 filters and a 3-wide kernel, motivated by the different n-gram frequencies between legitimate and illicit transactions.[6]

The third model is a CNN-variant called Temporal Convolution Network (TCN), which is more suited for sequential data (Yan et al. 2020). Here, the TCN receives a sequence of action (trainable) embeddings and applies 1D causal convolutions by convolving the output at time t only with outputs at time t and earlier. The output of the convolution is flattened and passed to a sigmoid activation. Table 7 presents the results of these three neural approaches. CNN is the best of the three in all metrics, yet worse than the best-performing SVM on top of TF-IDF features.

4.5 Machine learning with features based on RFM and NLP

We showed previously that an action sequence contains important and useful information, and that an SVM algorithm can capture and achieve promising performance. Although this approach works surprisingly well, outperforming the traditional RFM-based features which model user behavior and are frequently employed in literature, a question remains: what are the benefits of combining the two engineering approaches? To address this question, we trained and assessed our four machine-learning algorithms, by combining the two sets of features through concatenation. Table 8 presents the results.

Adding RFM-based features harmed the NLP-based SVM but benefited the NLP-based LR, RF, and kNN.[7] The performance improvement of the former was significant (more than fifteen units) and led to a score that was better than all other models in terms of AUPRC and F_{05}. From the results, we can see that the LSTM model achieves the highest F_2 score of 0.522, followed by the TCN model with a score of 0.483. The LR model has an F2 score of 0.496, showing that our models are generally performing well. The RF and

[5] We used one hidden layer with 64 hidden units, ReLU activations, and a classification layer with sigmoid activation.

[6] The encoded representation is flattened and passed through a FFNN using one hidden layer of 32 neurons and ReLu activation, and a classification layer with a sigmoid activation on top.

[7] The AUPRC score of the latter two was slightly reduced.

Table 7 Assessment of deep learning fraud classifiers with standard (AUPRC, F_1), online (F_{05}) and offline (F_2) measures (best in bold) using action embeddings

F:RFM	F:NLP	Method	**AUPRC**	$F_{0.5}$	F_1	F_2
□	Embeddings	LSTM	0.176	0.241	0.177	0.231
□	Embeddings	CNN	**0.210**†	**0.348**†	**0.273**†	**0.290**†
□	Embeddings	TCN	0.204	0.292	0.255	0.276

Bold values indicate the highest value of each column

Table 8 Assessment of machine and deep learning fraud classifiers with standard (AUPRC, F_1), online (F_{05}) and offline (F_2) measures (best in bold) using RFM-based and NLP (TF-IDF or Embeddings) features

F:RFM	F:NLP	Method	AUPRC	$F_{0.5}$	F_1	F_2
☑	TF-IDF	LR	**0.404**†	**0.535**†	0.467	0.496
☑	TF-IDF	RF	0.240	0.398	0.316	0.299
☑	TF-IDF	kNN	0.054	0.176	0.200	0.231
☑	TF-IDF	SVM	0.252	0.314	0.291	0.301
☑	Embeddings	LSTM	0.397	0.511	**0.498**†	**0.522**†
☑	Embeddings	CNN	0.375	0.428	0.439	0.452
☑	Embeddings	TCN	0.387	0.471	0.477	0.483

Bold values indicate the highest value of each column

Table 9 Assessment of all models in all metrics using traditional RFM, NLP, and anomaly detection-based features. In bold are the best results per column. A dagger indicates that the best overall is achieved in that column

	AUPRC	$F_{0.5}$	F_1	F_2
LR	**0.433**†	**0.501**†	**0.487**†	**0.520**†
RF	0.240	0.397	0.314	0.312
kNN	0.054	0.176	0.199	0.231
SVM	0.002	0.001	0.002	0.005
LSTM	0.427	0.443	0.469	0.474
CNN	0.43	0.48	0.485	0.490
TCN	0.429	0.469	0.473	0.482

Bold values indicate the highest value of each column

kNN models achieve considerably lower scores of 0.299 and 0.231, respectively, indicating that they may not be suitable for this particular task.

4.6 Can anomaly detection help?

Combining traditional, commonly-employed features to our TF-IDF representation yielded the best performance in all evaluation settings, standard, online, and offline. However, could we potentially further leverage this engineering approach by experimenting with features derived from unsupervised anomaly detection? This is a common approach in fraud detection applications because such features can help identify data points that deviate from normality. To address this question, we added the anomaly score output of an Isolation Forest to our combined (TF-IDF and RFM) feature set, and re-trained our machine learning and neural models.

Table 9 presents the results, where we see that the best performing LR improved in AUPRC, F_1 and F_2, but not in $F_{0.5}$, where the performance deteriorated. This shows the significance of the two additional evaluation measures we suggested, vis. $F_{0.5}$ and F_2, which can complement the fraud detection studies and allow model selection that fits the standards of the end user. Not surprisingly, all neural methods improved in all metrics, with CNN being the best neural method and close to the best-performing LR. It is typically expected to achieve better results with LSTMs in sequence modelling tasks, so this result is somewhat interesting. We believe the reason for this performance difference lies in the kernel size the CNN uses (three), which captures local patterns in a similar way to how

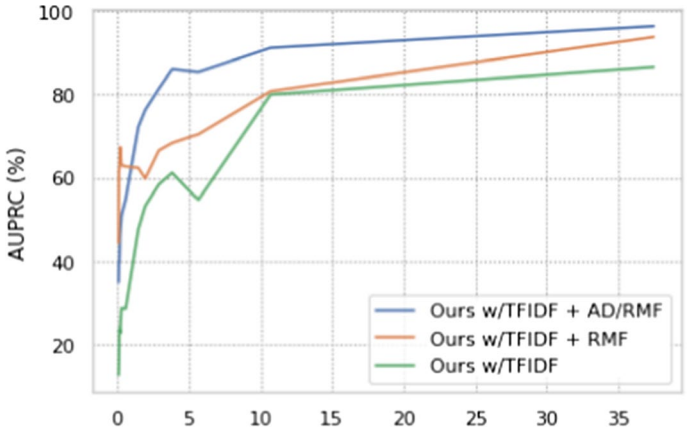

Fig. 4 Varying imbalance curve (VIC) of our LR on TF-IDF features (w/TF-IDF), with RFM-based features (+RFM) and with anomaly detection features added (+AD/RFM). The AUPRC is shown (Y-axis) by varying the percentage of fraudulent instances in the dataset (X-axis). From an extreme imbalance on the left to a more balanced fraction to the right

n-grams would. This shows that methods based on deep learning are promising, but more work is needed to allow their improvement. One such direction is transfer learning, which can be applied by using a much larger corpus to pre-train the action embeddings or the network weights (e.g., with self-supervision).

5 Discussion

Our results showed that an alternative, temporal, data engineering can leverage machine learning algorithms to outperform traditional feature engineering approaches. Verifying the intuition of Baesens et al. (2021), more complex approaches, such as dense representations and deeper neural networks, do not necessarily outperform their machine learning counterparts. When we use only TF-IDF features extracted from our transactional corpus (thus ablating RFM), or only traditional RFM-based features (ablating TF-IDF), the performance is considerably lower when using standard classification metrics. We argued also that evaluation should regard two specific scenarios in this task: the online, where fast alarming is required; and the offline, where Recall is at stake (i.e., detecting possibly missed cases). We suggested $F_{0.5}$ and F_2 and we used these to benchmark our models.

5.1 Studying the effect of a varying imbalance

As already stated, fraud detection is characterised by a large imbalance, with fraudulent transactions occurring in less than 0.1% of the data (see Sect. 3). Similarly to our work, other studies in literature make their own assumptions regarding the class balance, disregarding the realistic nature of the data and hence the problem. More importantly, the imbalance of the evaluation dataset affects the reported numbers, obfuscating the true performance of algorithms and making the potential of a fair comparison impossible to establish.

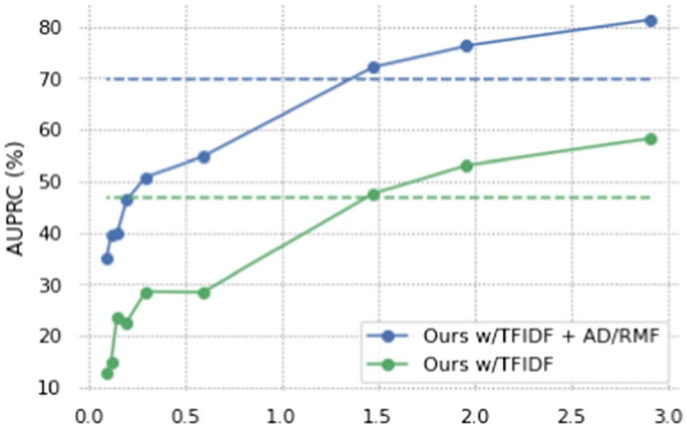

Fig. 5 The VICs of our vanilla LR w/TF-IDF (green) and our LR w/TF-IDF+AD/RFM (blue), as shown in Fig. 4 but zooming in on the extreme imbalance zone. In dashed lines the published performance of an LR with several RFM features (Baesens et al. 2021), w/o (green) and w/anomaly detection features added (blue). Our TF-IDF-based LR is on top at the 1.6% fraction (Color figure online)

In an attempt to adjust the class imbalance, we examined three data augmentation scenarios: (1) oversampling the fraudulent examples, (2) undersampling the legitimate examples and (3) doing both (1) and (2). Undersampling of the legitimate transactions was carried out in a simple manner, and a more detailed explanation of our oversampling approach follows.

For the classifiers that used TF-IDF features with RFM features, SMOTE (Chawla et al. 2002) was used to directly generate synthetic samples. However, for the classifiers that utilized user actions, we were able to use SMOTE only for the RFM features and had to use a different strategy to handle the sequential nature of the data. More specifically, we sampled, with replacement, pairs of fraudulent examples and cut each action sequence at a random point. We then concatenated the first part of one sequence with the second part of its pair to create a new one. We oversampled until we hit target class balances of 2%, 5%, 10%, 20%, and 50%, and trained on those additional fraudulent examples. Note that with our original class balance of 0.1%, these augmentation scenarios translated to 20, 50, 100, 200, and 500 times more fraudulent examples. None of those scenarios yielded better performance, which we attribute to oversampling from such a tiny number of examples. Furthermore, in the case where oversampled TF-IDF vectors were used, performance was significantly hindered by augmentation, which we believe happened due to the sparseness of the vectors.

Therefore, in order to study the performance of a fraud classifier across various possible imbalance settings, we suggest the use of a varying imbalance curve, or VIC in short. Such a curve can be made by training an algorithm on variants of the dataset whose class balance varies, ranging from an extreme imbalance to a more balanced setting. In our case, we yielded the latter by combining 100 legitimate and 66 fraudulent cases. Then, we started increasing the size of the legitimate transactions, until an imbalance of 1-to-1000 was hit. Figure 4 depicts the VIC of an LR algorithm trained on TF-IDF (green), with RFM-based features added (orange), and anomaly detection features added as well (blue). The score varies from the extreme imbalance (leftmost) to more balance settings on the right.

Inspection of the curves shows the problem is more challenging (lower AUPRC) at the leftmost side, where the extreme imbalance is assumed. On the other hand, more balanced

settings on the right lead to much higher AUPRC scores. Figure 5 shows the same curves for LR with TF-IDF features, with (blue) and without (green) added features, when we zoom in on the extreme imbalance zone (i.e., zooming in on the left of Fig. 4). The dashed lines correspond respectively to the performance reported in Baesens et al. (2021) for a 1.6% fraction of fraudulent instances, using an LR with several RFM features,[8] with (blue) and without (green) anomaly detection and other features. Their AUPRC was 46.80% without and 69.75% with the added features. For the same fraction (between 1.5 and 2), our TF-IDF-based LR is better and the same applies when we use the TF-IDF features with limited RFM and anomaly detection ones.

One hypothesis regarding why there are not any publicly available dataset for fraud detection on online actions, is that the available datasets comprise sensitive information, which is required by their systems in order to achieve satisfying performance. We question this hypothesis with this work, by releasing our data and code for public use.

6 Conclusions

Our work introduces an anonymised and publicly available dataset for online fraud detection, we demonstrate the potential of NLP methods for modeling online actions and achieving state-of-the-art results while preserving user privacy.

In order to assess model performance closer to a real-world business setting, we argued that standard evaluation metrics often employed in literature should be complemented with two more settings, one for online and one for offline evaluation. Online assessment regards the ability to early detect a suspicious transaction and can be measured with $F_{0.5}$. Offline assessment concerns the detection of all fraudulent instances and can be measured with the high-recall F_2. Investigating the results by using all our measures revealed that the addition of anomaly detection features improves all metrics except for the online scenario. This is an important observation for applications where online detection is at stake, indicating the significance of looking at both online and offline evaluation, along with standard metrics.

Finally, we observed that reporting of results in the literature is often based on different assumed imbalance settings. This harms comparability and hinders the progress of the field. To address this, we opted for a varying imbalance curve, which not only allowed us to investigate the task difficulty for various imbalance settings, showing that higher imbalance increases the challenge, but also to compare with the results reported in other published studies. While using the same machine learning algorithm, and with fewer traditional features compared to Baesens et al. (2021), the addition of NLP-based features led to a significant performance improvement. This improvement came not with the cost but with the benefit of privacy, given that we release our data for public use.

[8] We only used six RFM-based features and one based on anomaly detection (see Table 2), which is less than less than two times the features used in Baesens et al. (2021).

A Model parameter setting

A.1 Machine learning models

For the logistic regression model, we used L2 regularization with a penalty parameter of 1.0, and the LBFGS solver with a maximum of 100 iterations. For our k-Nearest Neighbors (k-NN) classifier, we used the k=2 and the KDTree algorithm from the scikit-learn library, while our random forest classifier was trained using 200 trees, a maximum depth of 50, and a minimum of 5 samples required to split an internal node. Finally, our support vector machine (SVM) model was trained using the radial basis function (RBF) kernel with a regularization parameter (C) of 1.0 and a polynomial kernel of degree 3.

A.2 Deep learning models

We built an LSTM model with 1 layer and 128 neurons. The model takes in a text input of maximum sequence length 256, as well as 6 additional numerical features. The text input is passed through an embedding layer before being fed into the LSTM layer. The last hidden state of the LSTM is concatenated with the additional numerical features, before passing through a dense layer with 64 units and a sigmoid activation function. The model is optimized using the Adam optimizer, with a learning rate of 0.001, and binary cross-entropy loss. We use accuracy, precision, recall, and AUC-PR as our evaluation metrics.

For our CNN we used an embedding layer with an output dimension of 32, and an input length of 150. This layer was followed by two 1D convolutional layers with 128 filters each and a kernel size of 2. We then applied batch normalization to the output of the convolutional layers and added a max pooling layer with a pool size of 4. The resulting output was flattened and concatenated with the other inputs. The concatenated output was then passed through four dense layers with 128, 96, 64, and 16 neurons, respectively. Finally, we used a sigmoid activation function in the output layer. The model was compiled with the Adam optimizer, binary cross-entropy loss, and Accuracy, Precision, Recall, and AUC as metrics.

The TCN model, similarly to the LSTM and CNN, takes two inputs, one for the action sequences and one for other variables. The action sequences are first processed through an embedding layer with an output dimension of 32, followed by a Temporal Convolutional Network (TCN) layer with 32 filters, 2 stacks, and skip connections. The other variables are passed through a separate branch in the model. The outputs of both branches are concatenated and passed through several fully connected layers with ReLU activations and dropout layers with a rate of 0.1. The final output layer uses a sigmoid activation function. The model uses a batch size of 128 and an Adam optimizer with a learning rate of 1e-4, and is compiled with binary cross-entropy loss and accuracy, precision, recall, and AUC as performance metrics.

Author contributions The first author undertook the engineering of the features, most of the experiments, and he contributed to authoring. The second author supervised the experiments, contributed to authoring and presentation. Also, he suggested the use of Natural Language Processing, the varying imbalance curves, and the evaluation measures. The third author assisted with the experiments and provided feedback. The fourth author provided feedback throughout the progress of this work.

Funding Open access funding provided by HEAL-Link Greece. Not applicable.

Data availability Our dataset will be released for public use upon acceptance.

Code availability Our code will be released.

Declarations

Conflict of interest Not applicable.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Achituve, I., Kraus, S., & Goldberger, J. (2019) Interpretable online banking fraud detection based on hierarchical attention mechanism. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)* (pp. 1–6). IEEE.

Baesens, B., Höppner, S., & Verdonck, T. (2021). Data engineering for fraud detection. *Decision Support Systems*. https://doi.org/10.1016/j.dss.2021.113492

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5*, 135–146.

Branco, B., Abreu, P., Gomes, A. S., Almeida, M. S. C., Ascensão, J. T., & Bizarro, P. (2020). Interleaved sequence RNNS for fraud detection. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. https://doi.org/10.1145/3394486.3403361

Carminati, M., Baggio, A., Maggi, F., Spagnolini, U., & Zanero, S (2018) FraudBuster: Temporal Analysis and Detection of Advanced Financial Frauds, pp. 211–233. https://doi.org/10.1007/978-3-319-93411-2_10

Carminati, M., Caron, R., Maggi, F., Epifani, I., & Zanero, S. (2015). Banksealer: A decision support system for online banking fraud analysis and investigation. *Computers & Security, 53*, 175–186.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority oversampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Fawcett, T., & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery, 1*, 291–316. https://doi.org/10.1023/A:1009700419189

Forough, J., & Momtazi, S. (2021). Ensemble of deep sequential models for credit card fraud detection. *Applied Soft Computing, 99*, 106883. https://doi.org/10.1016/j.asoc.2020.106883

Forough, J., & Momtazi, S. (2022). Sequential credit card fraud detection: A joint deep neural network and probabilistic graphical model approach. *Expert Systems, 39*(1), 12795. https://doi.org/10.1111/exsy.12795

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P.-E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications, 100*, 234–245. https://doi.org/10.1016/j.eswa.2018.01.037

Kovach, S., & Ruggiero, W. V. (2011). Online banking fraud detection based on local and global behavior. In *Proc. of the Fifth International Conference on Digital Society, Guadeloupe, France* (pp. 166–171).

Kunlin, Y. (2018). A memory-enhanced framework for financial fraud detection. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 871–874). https://doi.org/10.1109/ICMLA.2018.00140

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks,* (Vol. 3361(10)).

Liu, F. T., Ting, K., & Zhou, Z.-H. (2009). Isolation forest. In *2008 8th IEEE International Conference on Data Mining* (pp. 413–422). https://doi.org/10.1109/ICDM.2008.17.

Lucas, Y. (2019) Credit card fraud detection using machine learning with integration of contextual knowledge. Theses, Université de Lyon, Deutscheland, Universität Passau. https://tel.archives-ouvertes.fr/tel-02951477.

Mehana, A., & Nuci, K. P. (2020) Fraud Detection using Data-Driven Approach.

Nguyen, T.T., Tahir, H., Abdelrazek, M., & Babar, A. (2020). Deep Learning Methods for Credit Card Fraud Detection.

Panigrahi, S., Kundu, A., Sural, S., Majumdar, A.K., et al. (2009). Credit card fraud detection: A fusion approach using dempster-Shafer theory and Bayesian learning. *Information Fusion*, *10*(4), 354–363 . https://doi.org/10.1016/j.inffus.2008.04.001. Special Issue on Information Fusion in Computer Security.

Patel, Y., Ouazzane, K., Vassilev, V., & Li, J. (2019). Remote banking fraud detection framework using sequence learners. *Journal of Internet Banking and Commerce, 24*(1), 1–31.

Rinku, Narang, S. K., & Kishore, N. (2023). Issues in Credit Card Transactional Data Stream: A Rational Review. Lecture Notes in Networks and Systems (Vol. 421, pp. 775–789). www.scopus.com

Rodríguez, J. F., Papale, M., Carminati, M., & Zanero, S. (2022). A natural language processing approach for financial fraud detection. In *Proceedings of the Italian Conference on Cybersecurity ITASEC 2022, Rome, Italy*, June 20–23, 2022 (Vol. 3260, pp. 135–149). CEUR-WS.org.

Roy, A., Sun, J., Mahoney, R., Alonzi, L. P., Adams, S., & Beling, P. A. (2018). Deep learning detecting fraud in credit card transactions. In *2018 Systems and Information Engineering Design Symposium (SIEDS)* (pp. 129–134).

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one, 10*(3), 0118432–0118432. https://doi.org/10.1371/journal.pone.0118432

Wang, C. (2021). The behavioral sign of account theft: Realizing online payment fraud alert. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence* (pp. 4511–4618).

Wang, S., Liu, C., Gao, X., Qu, H., & Xu, W. (2017). Session-based fraud detection in online e-commerce transactions using recurrent neural networks. In Y. Altun, K. Das, T. Mielikäinen, D. Malerba, J. Stefanowski, J. Read, M. Žitnik, M. Ceci, & S. Džeroski (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 241–252). Cham: Springer.

Wedge, R., Kanter, J., Veeramachaneni, K., Moral, S., & Iglesias Pérez, S. (2019). Solving the false positives problem in fraud prediction using automated feature Engineering: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018. *Proceedings, Part III*, 372–388. https://doi.org/10.1007/978-3-030-10997-4_23

Yan, J., Mu, L., Wang, L., Ranjan, R., & Zomaya, A. Y. (2020). Temporal convolutional networks for the advance prediction of ENSO. *Scientific Reports, 10*(1), 1–15.

Zamini, M., & Montazer, G. (2018). Credit card fraud detection using autoencoder based clustering. In *2018 9th International Symposium on Telecommunications (IST)*, pp. 486–491. https://doi.org/10.1109/ISTEL.2018.8661129

Zhang, Z., Chen, L., Liu, Q., & Wang, P. (2020). A fraud detection method for low-frequency transaction. *IEEE Access, 8*, 25210–25220. (Cited By :10).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.