Int. J. Nonlinear Anal. Appl. 14 (2023) 5, 133–146
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2023.27720.3686

A new approach for detecting credit card fraud transaction

Cho Do Xuan[a,*], Dang Ngoc Phong[b], Nguyen Duy Phuong[b]

[a]Department of Information Security, Posts and Telecommunications Institute of Technology, Hanoi, Vietnam
[b]Department of Information Technology, Posts and Telecommunications Institute of Technology, Hanoi, Vietnam

(Communicated by Mouquan Shen)

Abstract

Nowadays, money transfer through the internet has become so popular because of its convenience and speed which makes users' lives easier. Even so, the safety of these transactions has been threatened by illegal activities causing great difficulty and loss for users. One of those unauthorized actions is fraud through credit cards used for financial transactions on online platforms. Therefore, research in detecting and early warning of fraudulent transactions through credit cards is essential today. In this paper, we propose a new approach for the task of early detection of fraudulent transactions based on a combination of two main methods, behavioral analysis techniques and supervised machine learning algorithms. Specifically, based on the behavioral analysis technique proposed in this paper, we have selected and extracted new features. These are features that have not been reported in previous studies. In addition, for the classification method, we propose to use a new advanced supervised machine learning algorithm, XGBoost. This is a newly researched and proposed machine learning algorithm. Based on the proposed approach in this paper, we have not only succeeded in synthesizing, analyzing and extracting the anomalous behavior of fraudulent transactions but also improved the efficiency of detecting suspicious transactions. Some experimental scenarios proposed in the paper have proven that our proposal in this paper is not only meaningful in terms of science but also in practical terms when the results of the paper have been proven more effective than some other approaches.

Keywords: Fraud detection, Anomaly behavior, Feature engineering, Class imbalance, Machine learning, XGBoost
2020 MSC: 68T05, 68T07

1 Introduction

Fraud is an offensive activity, carried out by an unauthorized person by cheating innocent people. Credit card fraud involves stealing essential credentials from the cardholder and using them in an unauthorized manner by the fraudsters either by using phone calls or SMS [1]. According to [21], data released on 16 December 2021 shows a 9.2% rise in fraud on payment card transactions in the 12 months to 30 June 2021 (FY21) alongside an increase in online spending during COVID-19 lockdowns. With total spending on cards rising 5.4% to $847.3 billion during the same period, the fraud rate in FY21 was 57.8 cents per $1,000 spent, up from 55.8 cents per $1,000 in FY20, but well below the rate of 73.8 cents in FY18. Fraudsters are inflicting huge financial and reputational losses on businesses and customers alike, using multiple avenues to monetize their exploits. Effective protection relies on accurately distinguishing between legitimate

*Corresponding author
Email addresses: chodx@ptit.edu.vn (Cho Do Xuan), phongdn@ptit.edu.vn (Dang Ngoc Phong), phuongnd@ptit.edu.vn (Nguyen Duy Phuong)

Received: July 2022 *Accepted:* March 2023

customers and fraudsters in real-time. Many techniques have been used to detect fraudulent transactions. [2] using fuzzy logic-based for E-transactional fraud detection, [10] proposes a novel hybrid method with dynamic weighted entropy for handling the problem of class imbalance with an overlap in credit card fraud detection. However, many researchers work across a variety of AI and machine learning areas, including deep neural networks, seeking the best solutions for the fraud prevention industry. Surveying recent research on detecting fraudulent transactions, we found that two issues need further attention and clarification in these studies, including fraudulent behavior and methods to detect them. Fraud detection systems that have limited in terms of proposed solutions, datasets, and difficulty in the experiments due to lacking knowledge of fraud schemes. Most researchers used classic machine learning algorithms, including deep learning, but these methods didn't work well on unbalanced fraud transaction datasets, which are the most common in the real world. To solve the above problems, in this paper we propose a feature engineering method to extract information based on combining the most relative features with fraud transactions. Moreover, we use an XGBoost model, a boosting machine learning algorithm instead of classic machine learning because it outperforms other methods when used on an imbalanced dataset.

The novelty and science suggested in the paper include:

• Proposed several features that exhibit the unusual behavior of fraudulent transactions. Accordingly, these behaviors in the article are all new features that have not been publicized in previous research.

• It is proposed to use the XGBoost algorithm for the task of classifying normal and abnormal transactions. The use of this algorithm has great scientific significance to help improve the efficiency of the process of classifying normal and abnormal transactions.

The presentation layout of the paper is as follows: in part 2, we present some research related to the problem of detecting and classifying fraudulent transactions. Part 3 is our method, in which section 3.1 describes the approach, section 3.2, 3.3 describes the method of extracting abnormal behaviors and lists these features in detail, and section 3.4 of the paper will detail the operating principle of the XGBoost algorithm. The experiment part and the evaluation of the model are presented in part 4 of the paper. The conclusion is presented in section 5.

2 Related works

Numerous Machine Learning techniques and methods have been designed and used in various experiment studies to predict fraud in credit card transactions. This topic has been investigated by a big number of researchers in recent years. [14] evaluates and compares the performance of 9 techniques - logistic regression, K-nearest neighbors, random forest, naive Bayes, multilayer perceptron, ada boost, quadrant discriminative analysis, pipelining, and ensemble learning on the credit card fraud data and uses accuracy, precision, recall, F1 score, and confusion matrix to compare the performance of these techniques in an attempt to find the most suitable one. Recently, many researchers have noted that ANNs perform significantly better on fraud prediction than discriminant and accounting ratio-based algorithms, as discussed in ([1, 14, 9, 12, 4, 19]). [1] used an artificial neural network (ANN) which gives accuracy approximately equal to 100% but they didn't use recall and precision scores with a skewed dataset so these results may not be accurate. [14] compared an Artificial Neural Network trained by the Simulated Annealing technique (SA-ANN) with a proposed emerging online learning technology in anomaly detection known as the Hierarchical Temporal Memory based on the Cortical Learning Algorithms (HTM-CLA). The results show that the maximum accuracy can be obtained from the SA-ANN technique. Thus, the HTM-CLA technique may not entirely be a better technique when compared with other fundamental neural network schemes. [4] proposed an approach for detecting statement fraud through the combination of information from financial ratios and managerial comments within corporate annual reports. They employed a hierarchical attention network (HAN) to extract text features and Analysis (MD&A) section of annual reports and the results demonstrate that textual features of MD&A sections extracted by HAN yield promising classification results and substantially reinforce financial ratios.

Many authors found that ensemble models and oversampling techniques outperform most machine learning linear models ([8, 15, 11, 17]). [17] used ensemble-based methods (random forest and gradient boosting) and deep neural networks. And the results indicated that the Random Forest model achieved significantly better performance than the standard Logistic Regression and other machine learning methods, as evidenced by the metrics of accuracy, precision, F1 Score, Cohen's Kappa, and MCC. Moreover, [13] compare various approaches for credit card fraud detection problems and they use homogeneous and heterogeneous Poisson processes to determine the probability of predicting fraud with the various intensity parametric functions. On the other hand, [13] used the RPT knowledge graph to detect financial fraud. The results show that the features derived from the RPTs knowledge graph contribute to improving financial fraud detection accuracy. [18] combine traditional features with knowledge graph models, and learn new

representations enriched with feature embedding of various financial categories. Experiment investigations with the five classical classifiers showed that the correlation information improved some of the classifiers' performance such as SVM, K-NN, and logistic regression. Also, the SVM method is more sensitive to correlation information than the rest of the methods. But it seems like tree-based and boosting machine learning models achieve high scores in detecting credit card fraud, detailed in ([6, 16, 5]). In [5] paper, Gradient boosting in combination with an autoencoder showed a high quality of classification. [6] used various machine learning methods including Decision Tree, and Random Forest and they used Matthews Correlation Coefficient metrics to deal with an imbalanced dataset. By applying the SMOTE, they observed that Logistic Regression, Decision Tree, and Random Forest are the algorithms that gave better results.

3 Fraud detection method D

3.1 Overview of proposed method

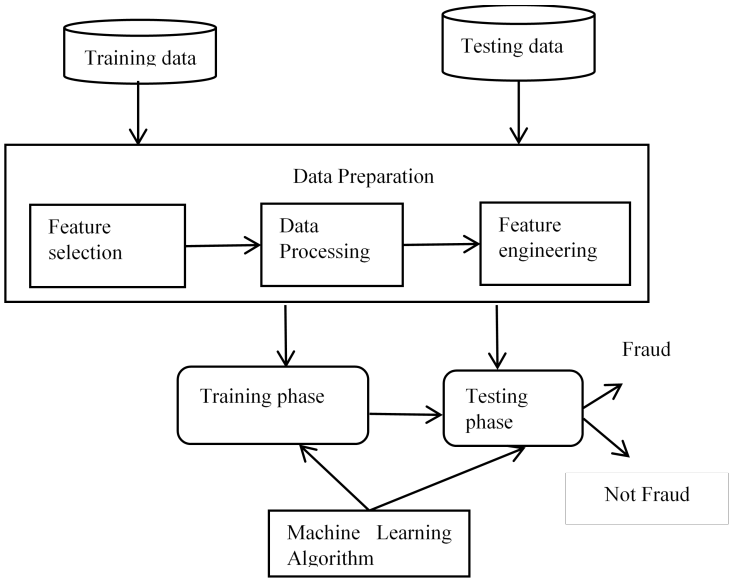

Figure 1: Model of fraud transaction detection based on machine learning

From Figure 1, the model will include the following main components:

• Training and Testing data: We use data from VestaCorporation. Regarding Vesta, our dataset includes timedelta from a given time reference datetime (not an actual timestamp), transaction payment amount in USD, payment card information, (such as card type, card category, issue bank, country, etc.), address, distance, Vesta engineered rich features, (including ranking, counting, and other entity relations).

• Data preparation phase: The goal of this phase is to standardize and extract data demonstrating the normal and unnormal behavior of fraudulent transactions. To do this, in this paper we propose several small steps including feature selection, data processing, and feature engineering. Details of this process are presented in section 3.2.2 of the article.

• Training phase: This is the stage of training the data to classify fraudulent transactions later. Accordingly, the abnormal behavior of fraudulent transactions after extraction will be analyzed and evaluated by machine learning algorithms to create a knowledge database. Details of this process will be presented in section 3.3.2 of the article.

• Testing phase: This is the prediction phase of abnormal transactions based on the knowledge database built in the training phase.

3.2 Feature engineering to extract the abnormal behavior of fraudulent transactions

3.2.1 Features overview

Table 1 lists all original features in the dataset.

Table 1: List of original features

No	Feature group	Feature name	Data type	Description
1		TransactionDT	numeric	Timedelta from a given reference datetime (not an actual timestamp). TransactionDT first value is 86400, which corresponds to the number of seconds in a day (60 * 60 * 24 = 86400) so the unit is seconds.
2		TransactionAmt	numeric	Transaction payment amount in USD
3		ProductCD	category	Product code, the product for each transaction
4	Address group	addr1	category	Billing region of purchaser
5		addr2	category	Billing country of purchaser
6		dist	category	Distances between (not limited) billing address, mailing address, zip code, IP address, phone area, etc.
7	Email domain	R_emaildomain	category	Purchaser email domain
8	group	P_emaildomain	category	Recipient email domain
9		card1	numeric	Payment card
10		card2	numeric	information,
11		card3	numeric	such as card
12	Card group	card4	category	type, card
13		card5	numeric	category, issue
14		card6	category	bank, country, etc.
15 - 28	Counting group	C1 - C14	numeric	Counting, the actual meaning is masked.
29 - 43	Timedelta group	D1 - D15	numeric	Timedelta, such as days between previous transactions, etc.
44 - 52	Match group	M1 - M9	category	Match, such as names on card and address, etc.
53 - 63	Identity group	$id_{01} - id_{11}$	numeric	Features for identity, which are collected by Vesta and security partners such as device rating, ip_domain rating, proxy rating, etc. Also it recorded behavioral fingerprints like account login times/failed to login times, how long an account stayed on the page, etc.
64		id_{12}	category	
65 - 66		$id_{13} - id_{14}$	numeric	
67 - 68		$id_{15} - id_{16}$	category	
69 - 74		$id_{17} - id_{22}$	numeric	
75		id_{23}	category	
76 - 78		$id_{24} - id_{26}$	numeric	
79 - 83		$id_{27} - id_{31}$	category	
84		id_{32}	numeric	
85 - 90		$id_{33} - id_{38}$	category	

91			isFraud	numeric	The logic of our labeling is to define reported chargeback on the card as fraud transaction (isFraud=1) and transactions posterior to it with either user account, email address or billing address directly linked to these attributes as fraud too. If none of the above is reported and found beyond 120 days, then we define it as a legit transaction (isFraud=0).
92	Device		DeviceType	category	User's device type
93			DeviceInfo	category	Information about user device
94 - 432	Vesta group	feature	V1 - V339	numeric	Vesta engineered rich features, including ranking, counting, and other entity relations.

In this dataset, we have a total of 432 features but not each of them was meaningful and necessary for our model to train on. Some features have lots of missing values (some had more than 90%) and we clean them by removing these features, this data cleaning process will be discussed in section 3.2.2. Moreover, the Vesta feature group, which has the most features in the dataset, all of these are numeric features and have missing values more or less. But they look similar when plotted in the graph so we can remove some of these features to reduce the size of the dataset. Details about this technique will be proposed in section 3.2.2.

3.2.2 Feature engineering method

Feature engineering is the most important step when dealing with data in any machine learning task. Normally, you do some data cleaning, find and remove features with the most missing data or replace the missing data with a value, etc. But in this case, we found a group of features in the same category, and most of them had missing values so we've been thinking that we can use these values to find similarities between them. And feature selection became my first step in this phase for possible dimensionality reduction in the dataset. Here is our technique, in this group of features, we split them into smaller groups with the same number of missing values. Then in each group, just assume we had a group like this [A, B, C, D, E, F], we used the Pearson correlation to find similarities between these features. Next, with each unit in the group (let's assume it was A), we took all the features in the remaining with similar scores greater than 0.75 to form a group and so on, the result might be looks like this [[A, C, D], [B, F], [E]]. Finally, for each group we have split, we selected a feature with the most unique values because these have high correlations so the one we chose should become the most important here. Using this technique, we had reduced one-third of the initial group but dropping these would not affect our model. In contrast, it decreased the memory usage and dimensionality of the features in the dataset. The second step in this phase is data processing, which includes data cleaning, encoding, and labeling. We removed the feature if it has more than 80% of missing values otherwise just filled it with -1. When we have a cleaned, corrected dataset, transforming category values to numerical by labeling them is necessary. After that, feature engineering was taken into account, what we did here is just combine two features and applied aggregate functions. Our challenge here is to predict fraudulent transactions meanwhile, the transaction amount was the first choice to be the aggregated value. We chose the group features based on feature importance values with the baseline model and the transaction amount. Then six aggregate functions (sum, mean, max, min, count, nunique) were used per combined pair. With categorical sets, we concatenated the training data with the test data then we counted the value frequency of each feature.

To write feature names easier, we have denoted them by shortening their name to avoid the new combined feature names becoming too long. Table 2 shows the feature names and their abbreviations. Table 3 lists the features selected for the engineering process.

3.2.3 Fraud detection using XGBoost

In this paper, we propose to use the XGBoost algorithm for detecting fraudulent transactions. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework [21]. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now. The XGBoost algorithm was developed as a research project at the [21]. Tianqi Chen and Carlos Guestrin presented their paper at SIGKDD Conference in 2016 and caught the Machine

Table 2: Feature symbol table

Group	Feature name	Symbol as
Card	card	c
Counting	C	C
Timedelta	D	D
Match	M	M
Identity	id	id
None	DeviceType	DT
	DeviceInfo	DI
	ProductCD	PCD
	P_emaildomain	Pe
	R_emaildomain	Re
	addr1	ad1
	addr2	ad2
	TransactionAmt	TA

Table 3: Engineered features table

Group	Feature list	Total
Card	card1 to card6	6
Counting	C1 to C14	14
Timedelta	D1 to D5, D10, D11, D15	8
Match	M1 to M9	9
Identity	id_12, id_15, id_16, id_23, id_27 to id_31, id_33 to id_38	15
None	DeviceType, DeviceInfo, ProductCD, addr1, addr2, P_emaildomain, R_emaildomain	7

Learning world by fire. Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications.

XGBoost is an ensemble tree method that applies the principle of boosting weak learners using the gradient descent architecture. In boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous trees. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals. The base learners in boosting are weak learners in which the bias is high, and the predictive power is just a tad better than random guessing. Each of these weak learners contributes some vital information for prediction, enabling the boosting technique to produce a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance. In contrast to bagging techniques like Random Forest, in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits. That said, XGBoost has become a widely used and really popular tool among Data Scientists in the industry, as it has been battle-tested for production on large-scale problems.

The [21] proposed the building principle as well as the predicting and optimizing of this algorithm. Based on those descriptions, we focus on how to make predictions with the following main steps:

Model Input: Dataset D: $D = (x_i, y_i), i = 1, 2, ...N$.

Step 1: Initialization: Make an initial prediction, by default is 0.5, so $p_0 = 0.5$ for all samples in the datasets. We can quantify how good the prediction is with a Loss function:

$$L(y_i, p(x_i)) = -[y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))]. \tag{3.1}$$

where: y_i is a observed value, p_i is a predicted value.

Step 2: Build trees and calculate optimal values: XGBoost uses the Loss function to build trees sequentially and make predictions based on all previous trees. For each tree (m) in total trees we set (M), we make prediction (p_m) by

Table 4: List features after engineering

Group	SubGroup Aggregated					
	Max	Min	Mean	Count	Sum	Nunique
Card	c1_TA_ max	c1_TA_ min	c1_TA_ mean	c1_TA_ count	c1_TA_ sum	c1_TA_ nunique
Counting	C1_TA_ max	C1_TA_ min	C1_TA_ mean	C1_TA_ count	C1_TA_ sum	C1_TA_ nunique
Timedelta	D1_TA_ max	D1_TA_ min	D1_TA_ mean	D1_TA_ count	D1_TA_ sum	D1_TA_ nunique
Match	M1_TA_ max	M1_TA_ min	M1_TA_ mean	M1_TA_ count	M1_TA_ sum	M1_TA_ nunique
Identity	id12_TA_ max	id12_TA_ min	id12_TA_ mean	id12_TA_ count	id12_TA_ sum	id12_TA_ nunique
None	DT_TA_ max	DT_TA_ min	DT_TA_ mean	DT_TA_ count	DT_TA_ sum	DT_TA_ nunique
	DI_TA_ max	DI_TA_ min	DI_TA_ mean	DI_TA_ count	DI_TA_ sum	DI_TA_ nunique
	PCD_TA_ max	PCD_TA_ min	PCD_TA_ mean	PCD_TA_ count	PCD_TA_ sum	PCD_TA_ nunique
	Pe_TA_ max	Pe_TA_ min	Pe_TA_ mean	Pe_TA_ count	Pe_TA_ sum	Pe_TA_ nunique
	Re_TA_ max	Re_TA_ min	Re_TA_ mean	Re_TA_ count	Re_TA_ sum	Re_TA_ nunique
	ad1_TA_ max	ad1_TA_ min	ad1_TA_ mean	ad1_TA_ count	ad1_TA_ sum	ad1_TA_ nunique
	ad2_TA_ max	ad2_TA_ min	ad2_TA_ mean	ad2_TA_ count	ad2_TA_ sum	ad2_TA_ nunique

calculating an output value for each leaf, b can be found by minimizing this equation.

$$\sum_{x_i \in R_j} L(y_i, p_{m-1}(x_i) + b_j) + \frac{1}{2}\lambda b_j^2. \tag{3.2}$$

where: R_j, $(j = 1, 2, ...J)$ is a single leaf that all value(s) $(x_i, i = 1, 2, ...N)$ fall into. b_j is an output value for a leaf in total leaves a tree has (J). λ is a number to scale the b value, the more emphasis we give the λ, the optimal b gets close to 0. p_{m-1} is a prediction made by the previous tree.

To find b value, we follow these steps:

- Compute the gradient for each value:

$$g_i = \frac{\partial L(y_i, p_{m-1}(x_i))}{\partial p_{m-1}(x_i)}, x_i \in R_j, \tag{3.3}$$

g_i is the first derivative of the Loss function and is called gradient.

- Compute the gradient for each leaf:

$$G_j = \sum_{g_i \in R_j} g_i. \tag{3.4}$$

- Compute the hessian for each value:

$$h_i = \frac{\partial^2 L(y_i, p_{m-1}(x_i))}{\partial p_{m-1}(x_i)^2}, x_i \in R_j, \tag{3.5}$$

h_i is the second derivative of the Loss function and is called hessian.

- Compute the gradient for each leaf:

$$H_j = \sum_{h_i \in R_j} h_i. \tag{3.6}$$

To minimize function (2), XGBoost used a Second Order Taylor Approximation to simplify the math when solving for the optimal output value b. The Loss function in (2) will can be approximated by the function below:

$$\sum_{x_i \in R_j} L(y_i, p_{m-1}(x_i) + b_j) = \sum_{x_i \in R_j} L(y_i, p_{m-1}(x_i)) + G_j b_j + \frac{1}{2} H_j b_j^2. \tag{3.7}$$

In (7), $\sum_{x_i \in R_j} L(y_i, p_{m-1}(x_i) + b_j)$ does not contain the output value b, that means they have no effect on the optimal output value, so we can remove them from the optimization. So the function (2) will become:

$$G_j b_j + \frac{1}{2} H_j b_j^2 + \frac{1}{2} \lambda b_j^2. \tag{3.8}$$

Now, to solve (8), we take the derivative with respect to the b value, set the derivative equal to 0 and solve for the b value. The result show below:

$$b_j = \frac{-G_j}{H_j + \lambda}. \tag{3.9}$$

We calculate the value for all the leaves by repeating these steps.

Step 3: Make final predictions:

$$p(x_i) = p_0 + \mu \sum_{m=1}^{M} b^m \tag{3.10}$$

where μ (XGBoost calls this value eta) is the learning rate to scale the prediction, b^m is the output value for a leaf that x_i fall into tree (m).

Model Output: Final predictions: $p(x_i)$ with i=1, 2, ..., N.

4 Experiments and evaluation

4.1 Experiment data

The dataset used in this paper was collected at IEEE-CIS Fraud Detection, Kaggle.com. The data comes from Vesta's real-world e-commerce transactions and contains a wide range of features from device type to product features. Table 5 below details the components of the dataset.

Table 5: Dataset information

Total sample	Fraud	Non-Fraud
590,540	20,663	569,877

From Table 5, it can be seen that there is a huge difference in the number of normal records and the number of fraudulent transactions. For such datasets, the training and classification tasks are very difficult. However, this is a meaningful and realistic dataset because it is common in the real world.

4.2 Experiment scenario

4.2.1 Preparing the dataset

During the experiment, the dataset will be randomly divided into 2 parts at the rate of 80% and 20%. In it, the larger dataset will be used for the training process. The other dataset will be used during testing and evaluation.

4.2.2 Training and evaluating scenarios

To evaluate the effectiveness of the proposed model, we will conduct two experiment scenarios as follows:

• Scenario 1: Experimentally evaluate the ability to detect fraudulent transactions using the XGBoost algorithm. During the experiment, we will fine-tune some parameters of the algorithm to test and evaluate the effectiveness of the algorithm when dealing with an imbalanced dataset. The experiment results of this scenario will not only clarify the efficiency of the algorithm but also find out which parameters the algorithm performs best.

• Scenario 2: Compare the XGBoost model in the paper with some other approaches including Random Forest, Logistic Regression, and Decision Tree. on the same experiment dataset. The results will show that XGBoost outperforms all these algorithms in terms of classifying fraudulent transactions.

4.3 Evaluation metrics

Confusion matrix: a table with four (with binary classification problems) different combinations of predicted and actual values.

Table 6: Confusion matrix

	Predicted fraud transaction	Predicted normal transaction
Real fraud transaction	TP	FN
Real normal transaction	FP	TN

Where:

• True positive (TP) is the number of fraud transactions correctly classified.

• False positive (FP) is the number of fraud transactions misclassified as normal transactions.

• True negative (TN) is the number of normal transactions correctly classified.

• False negative (FN) is the number of normal transactions misclassified as fraud transactions.

• Accuracy: the fraction of predictions the model got right.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

• Precision: refers to the number of corrected fraud transactions (TP) divided by the total number of fraud transactions predicted (TP + FP). High precision means the model is more accurate when predicting fraud transactions.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

• Recall: refers to the number of corrected fraud transactions (TP) divided by the total number of fraud transactions in the dataset (TP + FN). High recall means the rate of missing the true fraud transactions is low.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

• F1: is the harmonic mean of precision and recall. High F1 value means the classifier is good.

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.4}$$

• ROC AUC: ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. ROC is formed by calculating two values: True positive rate (TPR) and False positive rate (FPR).

$$TPR = \frac{TP}{TP + FN} \tag{4.5}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.6}$$

4.4 Experiment results

4.4.1 Experiment results of scenario 1

Table 7 below describes the experiment results of scenario 1.

Table 7: XGBoost results

N. of estimators	Scale	Acc	Pre	Rec	F1	ROC AUC
5000	28	0.990	0.891	0.819	0.854	0.908
5000	1	0.989	0.977	0.704	0.818	0.851
4000	1	0.946	0.359	0.694	0.473	0.825
4000	28	0.946	0.385	0.896	0.539	0.922

Table 7 result shows us that: Our model performs better with a higher number of estimators (0,99 and 0.989 accuracy with 5000 estimators) and the recall score on a scale of 27 is the best among these. The scale stands for scale_pos_weight, this is an XGBoostś hyperparameter designed to tune the behavior of the algorithm for imbalanced classification problems. By default, the scale_pos_weight hyperparameter is set to the value of 1.0 and has the effect of weighing the balance of positive examples, relative to negative examples when boosting decision trees. The negative class refers to the majority class (class 0) and the positive class refers to the minority class (class 1). In our case, the dataset is heavily skewed (97% majority class and only 3% minority class). That means our model likely overfit the majority class since it didn't have enough samples to predict the minority class.

The scale_pos_weight value has the effect of scaling errors made by the model during training on the positive class and encourages the model to over-correct them. In turn, this can help the model achieve better performance when making predictions on the positive class.

In this task, we set scale_pos_weight to 27 since the distribution between the majority class and minority class is approximately 27. This will give classification errors made by the model on the minority class (positive class) 27 times more impact, and in turn, 27 times more correction than errors made on the majority class.

Figure 2 shows the confusion matrix of the XGBoost model with 5000 estimators and scale_pos_weight equals 28 - the model with the best classification results.

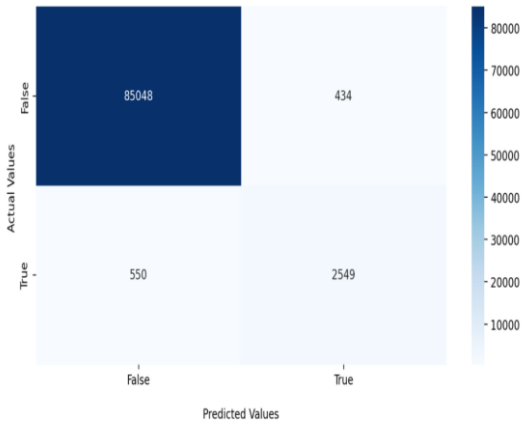

Figure 2: XGBoost's Confusion Matrix

The results show that the TP (2549) and FN (550) were the best of the four models and FP (434) is lower than the total actual positive points. That means our precision and recall are balanced and pretty high compared to other algorithms. We could achieve these scores because of XGB's scaling ability, it focused enough on the minority class but didn't overfit it and learned to predict by correcting the errors made by previous trees.

4.4.2 Experiment results of scenario 2

Table 8: Other machine learning method results

Approaches	Parameters	Acc	Pre	Rec	F1	ROC AUC
Random Forest [4, 12, 13]	n_estimators: 100; weights: {1: 1, 0: 1}	0.984	0.967	0.551	0.702	0.775
	n_estimators: 100; weights: {1: 27, 0: 1}	0.899	0.224	0.7637	0.346	0.8338
	n_estimators: 150; weights: {1: 27, 0: 1}	0.955	0.420	0.729	0.534	0.846
	n_estimators: 200; weights: {1: 27, 0: 1}	0.9	0.227	0.767	0.35	0.836
Logistic Regression [4, 17]	C: 1; weights: {1: 1, 0: 1}	0.966	0.725	0.058	0.107	0.529
	C: 1; weights: {1: 27, 0: 1}	0.754	0.096	0.719	0.17	0.737
	C: 0.01; weights: {1: 27, 0: 1}	0.754	0.096	0.719	0.17	0.737
Decision Tree [4, 10, 17]	max_depth: None; weights: {1: 1, 0: 1}	0.971	0.571	0.633	0.6	0.808
	max_depth: None; weights: {1: 27, 0: 1}	0.970	0.576	0.589	0.582	0.786
	max_depth: 15; weights: {1: 27, 0: 1}	0.892	0.213	0.771	0.333	0.834
	max_depth: 25; weights: {1: 27, 0: 1}	0.942	0.34	0.707	0.459	0.828

The experiment results in Table 8 show that:

With Random Forest (RF), we achieve a 0.984 accuracy score and 0.967 precision on the baseline but the recall score is pretty low (0.551) because the model was overfitting on the majority class. But when we scaled it to 27, the recall score was increased but overall, RF performs poorly on a skewed dataset and this is similar to Logistic Regression (LR) and Decision Tree (DT). With these two algorithms, even when we scaled it to 27, the results didn't change that much, recalls increased a little bit but precisions dropped dramatically. Especially LR, when we decreased the C value the result didn't change a bit. And because of the sensitivity of DT, the results changed a lot when we scaled it and adjusted max_depth but overall LR and DT are not a good choice when dealing with a heavily imbalanced dataset.

Figures 3, 4, and 5 show the confusion matrix of the RF, LR, and DT and we compared these results with XGBs to see why XGB is the good choice when dealing with a skewed dataset.

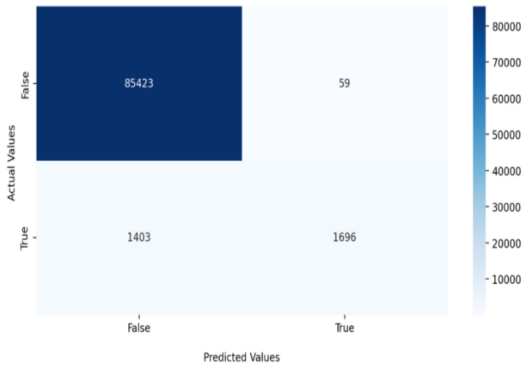

Figure 3: Random Forest's Confusion Matrix

With a baseline model, RF did a good job when the FP (96) is the lowest point of the four model's FP which is why its accuracy and precision are high. But with the recall score, RF only correctly predicted half of the total actual positive points. And even when we scaled it, the maximum recall score we reached was 0.767 and in contrast, the precision score dropped dramatically because it's a trade-off. So, RF performed better than LR and RT but it couldn't compare with XGB in terms of scaling the minority class.

The confusion matrix in Figure 4 shows that 870 out of 3099 Fraud transactions and 20906 out of 85482 Clean transactions were misclassified into the remaining class. This is a worse result because we got a super low precision (0.096) and the maximum recall is 0.719. Even the FP (20906) is six times more than the total actual Fraud (3099).

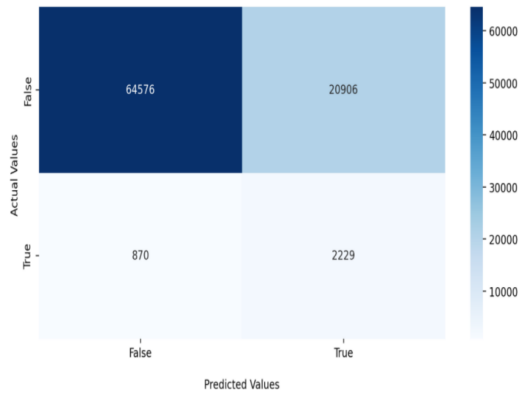

Figure 4: Logistic Regression's Confusion Matrix

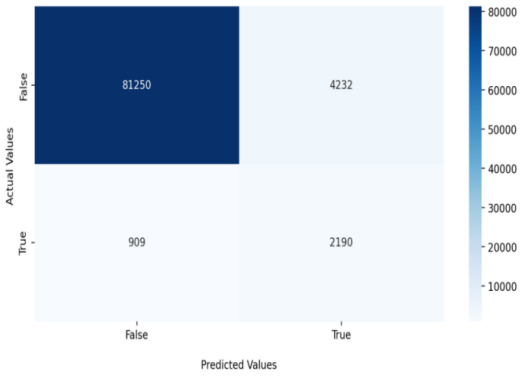

Figure 5: Decision Tree's Confusion Matrix

The DT did better than LR when the FP (4232) is smaller than LR's but it is still larger than the total actual positive point (3099). This is because the model was focused on the recall score since we scaled the minority class's weight 27 times more than the majority class to maximize the metric so it predicted more than the need. And this can't be a good model since both its precision and recall are low compared to RF and XGB.

4.5 Discussion

The results in Table 7 are better than in Table 8 with a 16.47% average higher than the other models, XGBoost did a great job at classifying the fraud transactions, the results show that all five metrics are high and there is not a too big difference between the scores. In contrast, the other algorithms not only have lower metrics than XGBoost but also their precisions and recalls are not balanced, and always have a big gap between them. So what makes XGBoost outperform the rest of these algorithms? It's because XGBoost is using a boosting method, boosting happens to be iterative learning which means the model will predict something initially and self-analyze its mistakes as a predictive point and give more weightage to the data records in which it made a wrong prediction in the next iteration. This process continues as a cycle. Hence technically, if a prediction has been done, there is at most surety that it did not happen as a random chance but with a thorough understanding and patterns in the data. Unlike Random Forest, at a high level, there are high chances that most of the trees could have made predictions with some random chances since each of the trees had its circumstances like class imbalance in our case whether we weighted the classes or not. In XG Boost, when the model fails to predict the anomaly for the first time, it gives more preferences and weightage to it in the upcoming iterations thereby increasing its ability to predict the class with low data points.

The XGBoost's confusion matrix in Figure 2 also gives the best results compared to other algorithms. It's TP (2549) and FN (550) were the highest of the four models while LG was unable to capture fraud transactions because its FP (20906) is too high so the precision is extremely low. DT was similar to LG but slightly better at precision and recall and RF was even better than those two when the FP (96) is the lowest point of the four models but RF only

correctly predicted half of the total actual positive points so it couldn't compare with XGB when predicting fraud transactions.

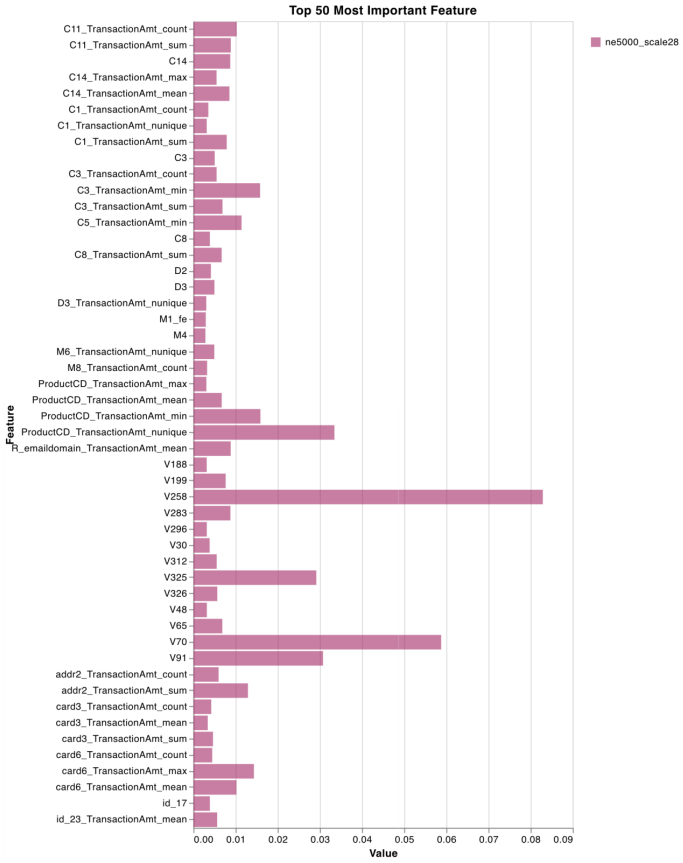

Figure 6: XGBoost's Feature Importance

Figure 6 below shows our XGBoost's feature importance for the training data, as you can see, most of the column names in the plot were processed features we created using our feature engineering method. That means our method had successfully created features that improved the performance of machine learning models, and optimal the fraud classification.

5 Conclusion

The proposed approach in the paper has improved the ability to accurately detect fraudulent transactions with an imbalanced dataset where there is a very large difference between the number of normal transactions and the number of fraud transactions. In particular, we have proposed a feature engineering method to create new attributes by combining the existing features. The experiment results in the article clearly show that these features play an important role in increasing the accuracy of the classification process. Obviously, with a difference of nearly 28 times between normal and fraud transactions, without a good and clear definition between abnormal behavior and normal behavior of transactions, it is not possible to bring about good results as in Table 7. In addition, the proposed using the XGBoost algorithm for the task of classifying fraudulent transactions is the right and reasonable choice. The Experiment results in scenario 2 have shown the outstanding efficiency of the XGBoost algorithm on all measures compared to other approaches and algorithms. In the future, based on the research results of this paper, we believe that it is necessary to improve the problem of anomaly detection in unbalanced datasets. This is a very important task not only to improve the efficiency of accurate detection of fraudulent transactions but also to reduce false alarms about normal transactions.

References

[1] R.B. Asha and K.R. Suresh Kumar, *Credit card fraud detection using artificial neural network*, Glob. Transit. Proc. **2** (2021), no. 1, 35–41.

[2] S.Md.S. Askari and Md. Anwar Hussain, *IFDTC4.5: Intuitionistic fuzzy logic based decision tree for E-transactional fraud detection*, J. Inf. Secur. Appl. **52** (2020), 102469.

[3] S. Bagga, A. Goyala, N. Gupta and A. Goyal, *Credit Card Fraud Detection using Pipeline and Ensemble Learning*, Procedia Computer Sci. **173** (2020), 104–112.

[4] P. Craja, A. Kim, and S. Lessmann, *Deep learning for detecting financial statement fraud*, Decision Support Syst. **139** (2020), 113421.

[5] J. Domashova and O. Zabelina, *Detection of fraudulent transactions using SAS Viya machine learning algorithms*, Procedia Comput. Sci. **190** (2021), 204–209.

[6] V.N. Dornadula and S. Geetha, *Credit card fraud detection using machine learning algorithms*, Procedia Comput. Sci. **165** (2019), 631–641.

[7] A. Izotova and A. Valiullin, *Comparison of Poisson process and machine learning algorithms approach for credit card fraud detection*, Procedia Comput. Sci. **186** (2021), 721–726.

[8] S. Kaisar and A. Chowdhury, *Integrating oversampling and ensemble-based machine learning techniques for an imbalanced dataset in dyslexia screening tests*, ICT Express **6** (2022).

[9] E. Kim, J. Lee, H. Shin, H. Yang, S. Cho, S.-k. Nam, Y. Song, J.-a Yoon, and J.-il Kim, *Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning*, Expert Syst. Appl. **128** (2019), 214–224.

[10] Z. Li, M. Huang, G. Liu, and C. Jiang, *A hybrid method with dynamic weighted entropy for handling the problem of class imbalance with overlap in credit card fraud detection*, Expert Syst. Appl. **175** (2021), 114750.

[11] M.J. Madhurya, H.L. Gururaj, B.C. Soundarya, K.P. Vidyashree, and A.B. Rajendra, *Exploratory Analysis of Credit Card Fraud Detection using Machine Learning Techniques*, Glob. Transit. Proc. **11** (2022).

[12] K. Maka, S. Palani Rajan, and S. Mallapur, *Selection of most significant variables to detect fraud in financial statements*, Materials Today: Proc. **7** (2020).

[13] X. Mao, H. Sun, X. Zhu, and J. Li, *Financial fraud detection using the related-party transaction knowledge graph*, Procedia Comput. Sci. **199** (2022), 733–740.

[14] E.N. Osegi and E.F. Jumbo, *Comparative analysis of credit card fraud detection in Simulated Annealing trained*, Artific. Neural Network Hierarch. Temporal Memory **16** (2021), 100080.

[15] Ch.Md. Rakin Haider, A. Iqbal, A. Hasan, and R.M. Sohel Rahman, *An ensemble learning based approach for impression fraud detection in mobile advertising*, J. Network Comput. Appl. **112** (2018), 126–141.

[16] I. Sadi Gali, N. Sael, and F. Benabbou, *Performance of machine learning techniques in the detection of financial frauds*, Procedia Comput. Sci. **148** (2019), 45–54.

[17] M.K. Severino and Y. Peng, *Machine learning algorithms for fraud prediction in property insurance: Empirical evidence using real-world microdata*, Machine Learn. Appl. **5** (2022), 100074.

[18] Y. Shen, C. Guo, H. Li, J. Chen, Y. Guo, and X. Qiu, *Financial feature embedding with knowledge representation learning for financial statement fraud fetection*, Procedia Comput. Sci. **187** (2021), 420–425.

[19] A. Shiraz Hashmi and T. Ahmad, *GP-ELM-RNN: Garson-pruned extreme learning machine based replicator neural network for anomaly detection*, J. King Saud Univer.Comput. Inf. Sci. **34** (2019), no. 5, 1768–1774.

[20] AusPayNet, *AusPayNet reminds consumers and merchants to be vigilant online as latest card fraud figures released*, 2021.

[21] T. Chen and C. Guestrin, *XGBoost: A scalable tree boosting system*, arXiv:1603.02754v3 (2016).