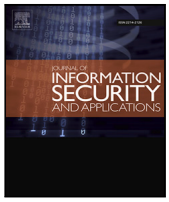
Journal of Information Security and Applications 78 (2023) 103618

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa



An ensemble learning approach for anomaly detection in credit card data with imbalanced and overlapped classes

Md Amirul Islam [a], Md Ashraf Uddin [b,*], Sunil Aryal [b], Giovanni Stea [a]

[a] Department of Information Engineering, University of Pisa, Italy
[b] School of Information Technology, Deakin University, Australia

ARTICLE INFO

Keywords:
Anomaly detection
Credit card
Ensemble
Meta-learning
Base learner
Classification

ABSTRACT

Electronic payment methods have become increasingly popular for business transactions, both online and in-person, across the globe. Anomalies like online fraud and default payments, which can result in substantial financial losses, have become more common as the usage of credit cards in online purchases has increased. To address this issue, researchers have explored various machine learning models and their ensemble techniques for detecting anomalies in credit card transaction data. However, detecting anomalies in this data can be challenging due to overlapping class samples and an imbalanced class distribution. Therefore, the detection rate of anomalies from minority class samples is relatively low, and general learning algorithms can be biased towards the majority class samples. In this paper, we propose a model called Credit Card Anomaly Detection (CCAD) that leverages the base learners paradigm and meta-learning ensemble techniques to improve the detection rate of credit card anomalies. We utilize four outlier detection algorithms as base learners and XGBoost algorithm as meta learner in the proposed stacked ensemble approach to detect anomaly in credit card transactions. We apply stratified sampling technique and k-fold cross-validation process to address the issues of data imbalance and overfitting. In addition, the discordance rate is calculated to enhance the accuracy of ensemble learning performances. The proposed model is trained and tested using two datasets: CCF (Credit Card Fraud) and CCDP (Credit Card Default Payment). Experimental results demonstrate that our approach outperforms existing approaches, particularly in detecting anomalies from the minority class instances of these datasets.

1. Introduction

Credit cards have become very common in the modern day for carrying out routine financial activities including commercial dealings and real-time services. Consumers make purchases using Internet-connected electronic devices, such as tablets, PCs, and smartphones. However, the potential of fraudulent transactions resulting in significant financial losses equal to billions of dollars every year has increased due to the increasing use of various online payment systems and electronic banking [1,2].

A report from [3] reveals a global loss of 28.58 USD billion in 2020 due to credit card fraud, affecting cardholders, consumers, and merchants worldwide. Furthermore, [4] found that fraudulent credit card transactions alone caused an 11 USD billion loss in the United States in 2020. The situation is alarming as a recent report [5] indicates a tripling of global credit card fraud over the last decade, with losses soaring from 9.84 USD billion in 2011 to 32.39 USD billion in 2021. The report predicts that enterprises worldwide could suffer cumulative losses of approximately 408.50 billion over the next ten years. Notably, the Malaysian banking sector faced substantial losses, recording a total loss of RM 51.3 million in fraudulent credit card transactions in 2016 [6,7]. Furthermore, 12.8% of credit cardholders in Malaysia struggled to meet the minimum balance payment, raising concerns for the Malaysian government.

Financial organizations need to address several issues related to credit cards, including (i) the issue of card owners neglecting refunds leading to default payments, and (ii) the presence of unauthorized third parties causing frauds. An automated anomaly detection system can significantly contribute to the resolution of this issue.

Over the years, researchers have actively sought solutions to the challenges posed by banking data, such as credit card fraud and default payment. Data mining has emerged as a promising approach in handling these issues. However, banking data presents unique challenges for data mining experts due to the following properties commonly

* Corresponding author.
E-mail addresses: mdamirul.islam@phd.unipi.it (M.A. Islam), ashraf.uddin@deakin.edu.au (M.A. Uddin), sunil.aryal@deakin.edu.au (S. Aryal), giovanni.stea@unipi.it (G. Stea).

https://doi.org/10.1016/j.jisa.2023.103618

Available online 7 October 2023

observed in the data: (i) overlapping of class samples (ii) imbalanced class distribution.

In credit card datasets, important categories, like fraud and default payments, are typically underrepresented. Learning algorithms can effectively identify and learn patterns from these classes when they have a sufficient number of records. However, the task becomes more challenging when the number of samples in these classes is small, making it difficult to generalize and define their decision boundaries accurately using traditional learning algorithms [1,2]. The situation becomes more complex when there is a significant overlap in attribute values among many normal transactions. In cases where the minority classes are linearly separable, the performance of the learning algorithm may not be significantly affected, even if the dataset is highly imbalanced [8,9].

Various research works have explored supervised machine learning approaches to deal with the issues of detecting fraud credit card. Most popular algorithms include SVM (Support Vector Machine) [10], DT (Decision Tree) [11], ANN (Artificial Neural Network) [12], LR (Logistic Regression) [10], LSTM (Long Short Term Memory) [13], and CNN (Convolutional Neural Network) [14]. Some researchers employed ensemble model to address the low rate of anomaly detection rate. The most used current ensemble approaches in this domain are bagging classifiers based on the decision tree [15], majority voting, AdaBoost with standard machine learning algorithms [16], and deep learning models [17]. However, the effectiveness of supervised learning is contingent on a sufficient number of instances from all classes, including normal and abnormal data. In the case of credit card datasets, there are still insufficient fraud data to train supervised models.

Further, the accuracy rate of anomaly detection in credit card transactions is often lower due to the complex and dynamic nature of the dataset. As a result, single supervised machine learning classifiers frequently struggle to learn the intricate characteristics inherent in credit card transactions. Most existing approaches predicts anomaly solely using single classifier, which did not become effective for anomaly detection in credit card transactions with higher accuracy. Alternatively, ensemble techniques have the potential to outperform individual models. Recognizing the advantages of this technique, we have developed a stacked ensemble model called the “Credit Card Anomaly Detection (CCAD)” model. This model aims to address the issue of low anomaly detection rates in the datasets utilized in this study.

In real-life scenarios, the occurrence of normal credit card transactions is higher compared to abnormal transactions. Therefore, a semi-supervised model, such as outlier detection, is more suitable for building an effective anomaly detection system. In a semi-supervised approach, we only need legitimate credit card transactions for training. The model identifies fraudulent transactions by recognizing deviations from the usual patterns of legitimate activities. This method addresses the issue of data imbalance and is more apt for real-world scenarios where gathering a substantial number of anomalous credit card transactions is difficult. Taking this into consideration, we have employed four outlier detection algorithms and combined them in an ensemble approach to develop a robust anomaly detection system.

To evaluate the effectiveness of our proposed approach, we utilized stratified cross-validation and compared it with state-of-the-art ensemble and individual models. The results demonstrated that our suggested model outperforms the existing models in the field of credit card anomaly detection. Our contributions to this work can be summarized as follows:

i. To propose a stack ensemble learning model to detect anomalies in credit card transactions. The model integrates semi-supervised outlier techniques and supervised machine learning algorithms. Our approach detects the credit card transaction as genuine or anomalous.
ii. To implement the model using stratified cross validation to tackle the issues of the imbalanced dataset and overfitting. Grid search hyper parameter tuning is applied to improve anomaly detection performance. We analyze the performance of the proposed model in terms of accuracy, precision, recall, F1-score, and AUC.
iii. To ensure the integrity of the ensemble model, we computed the discordance rate among the base learner models. This step aimed to ensure that the ensemble model is trained on distinct meta-features rather than relying solely on the predictions of different base-level models. By doing so, we minimized the risk of creating a biased model and ensured a more reliable and unbiased ensemble model.

Paper Outline. The remaining part of this research paper is structured as follows. In Section 2, related works is reviewed and proposed approach discussed in Section 3. Section 4 discusses the simulation results of proposed approach and comparisons with existing works. Finally, the conclusion and future research directions are summarized in Section 5.

2. Related work

Researchers and stakeholders have been exploring for effective strategies to address credit card fraud as the use of electronic payment systems, especially credit cards, continues to expand around the world. There needs to enhance existing fraud detection techniques and establish a more secure payment system as the characteristics of credit card attacks evolve. In this section, we describe recent state-of-the-art works to effectively mitigate electronic credit card fraud.

Alfaiz et al. [18] utilized a real-world credit card fraud dataset collected from a European bank to identify the fraud. The dataset is highly imbalanced with 31 features and contains only 492 frauds (0.173%) out of 284,807 online transactions. First, the authors trained nine ML algorithms and selected the top three performing ML algorithms. Next, resampling techniques was used for the selected ML algorithms. They showed that K-Nearest Neighbors (KNN) outperformed CatBoost method while detecting fraudulent transactions.

Malik et al. [19] presented a new hybrid machine learning architecture to detect frauds from the electronic credit card dataset. They used the IEEE Computational Intelligence Society (IEEE-CIS) [20] dataset, which was provided by Vesta Corporation. The dataset has 432 features and half a million credit card transactions, but the data are highly imbalanced, and the imbalance rate is 0.035. The authors trained and tested seven hybrid machine-learning models using the datasets to identify fraud. In their model, they first measured the performance of individual model and then hybrid methods were constructed based on the performance of single algorithm. They found that Adabost+LGBM is the best hybrid model to detect fraud. The authors did not measure the performance using the cross-validation process. As a result, their performance might vary significantly as the training and testing samples are changed.

Zhang et al. [21] presented an optimized anomaly detection model by dealing with an imbalanced issue in the credit card fraud dataset. The authors compared the experimental results of their data balancing techniques with synthetic minority oversampling technique (SMOTE). They claimed that their approach perform better than SMOTE to detect fraud. First, the authors used Isolation Forest (IForest) to detect fraud behavior accurately. Next, AdaBoost and one-class support vector machine (OCSVM) were utilized to improve the accuracy in detecting the outliers. Cross validation approach to evaluate the result was not employed in this work too.

Carcillo et al. [22] developed a hybrid model by combining supervised & unsupervised techniques to enhance the performance of the fraud detection system. The authors tested their model using real and annotated datasets of fraudulent identification. The limitation of this work is that data imbalanced issue was not handled.

Zhang et al. [23] employed an advanced feature engineering process and Homogeneity-Oriented Behavior Analysis (HOBA) for an electronic credit card fraudulent transaction detection system. The authors trained

and tested their model using a dataset collected from a bank in China. Imbalanced issue of dataset was not addressed in their works.

Karthik et al. [24] combined bagging and boosting ensemble learning techniques to accurately detect fraud credit cards. The research used UCSD-FICO competition credit card dataset & Brazilian bank dataset. The authors used a hybrid-level approach to tackle the imbalanced problem of these two datasets.

Forough et al. [25] used deep recurrent neural networks and novel voting techniques to construct an ensemble system to detect credit card fraud. They used two real-world datasets: 2013 European credit cards and Brazilian credit card datasets to train and test their model.

Asha et al. [26] used artificial neural network (ANN), support vector machine (SVM), and k-nearest neighbors (KNN) algorithms to detect and predict fraudulent transactions. They did not address the issue of imbalanced data and did not compare their models with other single classifiers.

Seera et al. [27] used an actual credit card fraud dataset obtained from a Australian bank to detect the payment card fraud. The dataset is highly imbalanced with 16 features. The authors applied 13 statistical machine-learning techniques to identify fraudulent transactions. They conducted a statistical hypothesis test to determine if aggregated features were distinguishable and if the genetic algorithm performed better than the original features at detecting fraudulent transactions.

Itoo et al. [28] utilized a skewed dataset to compare and analyze three ML algorithms: Naïve Bayes, KNN, and Logistic Regression (LR). The authors found that the LR model had achieved better results. However, the authors did not handle the imbalanced data. Olowookere et al. [29] utilized the same dataset utilized as [18]. The authors combined cost-sensitive meta-learning and ensemble learning paradigms to increase the fraud detection rate from an imbalanced dataset in their proposed framework. For the base classifiers, they used MLP, KNN, and DT machine learning algorithms.

Khatri et al. [30] utilized the European credit card fraud dataset (ECCFD) to compare some established supervised algorithms to distinguish among fraud and non-fraud. This dataset was made by ULB (*Université Libre de Bruxelles*) machine learning group & Worldline and contained 28 features. The authors found that the DT and KNN model performed better than other established models.

Kalid et al. [31] used Multiple Classifier Systems (MCS) to classify the fraud and non-fraud transactions because data inconsistencies reduced the fraud detection rate. The authors employed a sequential decision technique with MCS to increase the delectation rate of fraud. They used two datasets to justify their proposed model: Credit card fraud (CCF) and Credit card default payment (CCDP).

Dornadula et al. [32] used various kinds of ML classifiers to detect fraud and utilized the same dataset utilized as [18]. The authors successfully presented that RF classifiers had gained better results than other models.

Sohony et al. [33] utilized the AdaBoost classifier as a meta classifier and Na´ıve Bayes as a base classifier on the highly imbalanced credit card dataset with only 0.173% transactions of fraud to detect the anomaly. Data imbalanced issue was not handled.

Randhawa et al. [34] proposed a hybrid model by combining majority voting and AdaBoost, and they utilized the same dataset utilized as in [18]. The authors first utilized NB and ANN as the base classifier for the AdaBoost and parallel used majority voting to obtain the experiment's final results. Although authors demonstrated higher accuracy, recall and TN of their model, they did not show precision which might be impacted by the data imbalanced issue.

Ram et al. [35] utilized a deep Convolutional Neural Network (CNN) model to detect fraud.

Xenopoulos et al. [36] presented an ensemble learning model to detect fraud and utilized the same dataset utilized as [18]. Firstly authors used bootstrapped on the imbalance dataset such that every generated dataset has a balanced class distribution. After that, the authors applied a Deep Belief Network ensemble to each bootstrapped data.

Tiwarekar et al. [37] presented a hybrid model that utilized the Decision Tree (DT) model to aggregate Luhin's and Hunt's algorithms. Luhn's algorithm validates the credit card number, and address-matching rules are used to check the correct billing and shipping address. If billing and shipping addresses are matched, then its a high chance of being non-fraud; otherwise, it will be a fraud. However, the authors were not concerned about the performance evaluation of the model.

Xia et al. [38] utilized XGBoost, also known as fine-tuned boosting ensemble method, to resolve the classification issues. In their experiment, they used the Taiwan credit card default payment dataset and gained an accuracy of 69.36% and an AUC of 87.90%. However, the authors did not address the problem of oversampling class samples and unbalanced data, but the authors recommend that MCS be integrated with the XGBoost to improve the performance of the classification problem.

Singh et al. [39] presented an ensemble learning model of classification for credit scoring. In their work, they used the same dataset as in [38]. For the base classifier, they utilized bagging with a random forest classifier, and the dataset is imbalanced with 72% non-default payment and 28% default payment.

Charleonnan et al. [40] also presented a hybrid model that merged RUS and MRN algorithms to identify credit card fraud. In addition, the authors used three ML classifiers as base classifiers of the proposed model: Multilayer Perceptron (MLP), Radial Basis Function (RBF), and Naïve Bayes (NB). The performance evaluation was done on the Taiwan credit card default payment dataset and achieved an accuracy of 77.8%, sensitivity of 53.36%, and specificity of 88.13%.

Venkatesh et al. [41] presented an ensemble learning approach to predict the credit card defaulter. The authors used NB, RF, and bagging techniques in their model and used the same dataset utilized as in [38] for conducting the experiment. Moreover, they applied the CFS method to minimize the dataset's dimension to enhance classification accuracy. However, the authors did not conduct robust performance analysis and did not present the result of many important metrics except TPR or recall.

Table 1 summarizes recent studies with limitations for credit card anomaly detection. We provide the performance of the existing works in Table 1.

By observing and analyzing all the recent works mentioned above, we have found that most literature did not address the imbalanced issue of credit card datasets used in their works. There needs to develop a model which performance is not affected by the imbalanced issue of the datasets. Collecting enough number of abnormal credit card transactions is challenging. Using statistical methods like SMOTE, balancing malicious credit card data is not practical solution and the model built on such data balancing techniques fail to detect malicious credit card transactions in real life. Further, most current works are complex and are not enough time efficient in identifying anomalies. We require to develop a model that might be both space and time efficient through addressing class imbalanced issue. We also noticed that few researchers attempted to build an ensemble learning model of one class classifiers as base learner and supervised learning as meta learning for detecting credit card anomaly. We aim to explore such hybrid model of one class classifier and supervised classifier in detecting malicious credit card transaction where OCC can handle the scarcity of malicious credit card transactions.

3. Proposed architecture for credit card fraud identification

Anomaly identification in the context of credit card transactions involves modeling past transactions and learning to identify anomalies. The target of our work is to detect if a new credit card transaction might be fraudulent. In this work, we develop a model that can accurately detect fraudulent or anomalous transactions to prevent financial losses.

Table 1
Related literature reviews on ensemble learning and anomaly detection.

Ref.	Year	Method	Contribution	Transaction data	Evaluation metric	Metric value	Limitation
[18]	2022	K-Nearest Neighbors (KNN), KNN-CatBoost	Enhanced credit card fraud detection model	2013 European credit card fraud data [42]	AUC, Recall, F1-score	96.94%, 95.91%, 87.40%	Not enough validation and performances metrics.
[19]	2022	AdaBoost, Light Gradient Boosting Machine (LGBM)	Fraud detection in credit card data	IEEE-CIS [20]	Precision, ROC, Recall, F-Measure	97.00%, 82.00%, 64.00%, 77.00%	Did not handle the imbalanced data and did not focus on cross-validation.
[21]	2022	AdaBoost, One-class support vector machine (OCSVM)	Anomaly detection in credit card data	Credit card fraud dataset	Accuracy, Precision, Recall, F1-score	96.00%, 97.00%, 96.00%, 98.00%	Cross validation metric is not implemented and performances are not enough.
[22]	2021	Hybrid approach	Credit card fraud detection	Annotated and real datasets	Accuracy, Precision, Recall	97.20%, 98.30%, 95.00%	Did not handle imbalanced data and Cross validation metric is not implemented.
[23]	2021	Homogeneity-oriented behavior analysis (HOBA)	Credit card fraud detection	Extensive dataset of a bank in China	Precision, Recall, F1-score, AUC	96.00%, 94.50%, 96.02%, 97.10%	Performances are not enough and cross validation metric is not implemented.
[24]	2021	Adaboost, Random Forest, and Extra Trees	Credit card fraud detection	Brazilian bank data and UCSD-FICO competition credit card data	Accuracy, Recall, AUC	99.18%, 99.50%, 97.08%	Did not concern about precision metric and cross validation metric is not implemented.
[25]	2021	Sequential modeling-based ensemble model	Credit card fraud detection	European cards and Brazilian credit cards dataset	Precision, Recall, F1-score, AUC-ROC, AUC-PR	95.69%, 66.74%, 78.13%, 83.37%, 63.53%	Not enough performance and did not focus to overcome the imbalanced data.
[26]	2021	Artificial neural network (ANN), k-nearest neighbors (KNN), and a support vector machine (SVM)	Fraud detection from credit cards data	Kaggle dataset	Accuracy, Precision, Recall	93.49%, 97.43%, 89.76%	Did not focus to overcome the imbalanced data and performances are not enough.
[27]	2021	13 Statistical and machine learning models	Payment card fraud detection	Australian dataset	Accuracy, AUC, MCC	96.49%, 93.70%, 96.40%	Did not tackle imbalanced data and performances are not enough.
[28]	2021	K-nearest neighbor (KNN), Naive Bayes (NB), Logistic regression (LR)	Classify the fraud and non-fraud transactions of the credit card dataset	Skewed dataset	Accuracy, Precision, Sensitivity, F1-score	95.00%, 97.00%, 89.00%, 91.00%	Did not focus to tackle imbalanced data and performance is not enough.
[29]	2020	K-Nearest Neighbors (KNN), Decision Tree (DT) and Multilayer Perceptron (MLP)	Detection of credit card fraud	2013 European bank data [42]	ROC curve (AUC)	97.40%	Not enough validation metrics.
[30]	2020	Decision Tree (DT), K-Nearest Neighbors (KNN)	Credit card fraud detection	European credit card fraud dataset (ECCFD)	Precision, Recall	79.21%, 85.11% and 81.19%, 91.11%	Cross-validation metric is not implemented, and did not consider using any resampling techniques to overcome the imbalanced issues
[31]	2020	Multiple Classifier	Anomaly detection in credit card data	Credit card fraud (CCF) [42] and Credit card default payment (CCDP)[43]	Accuracy, Precision, Recall, AUC	99.90%, 83.45%, 87.20%, 87.90% and 93.00%, 95.5%, 84.08%, 70.30%	Not enough performance metrics are measured and did not implement cross-validation metrics.
[32]	2019	Multiple Classifier	Fraud detection from credit card dataset	2013 European credit card dataset [42]	Accuracy, Precision, MCC	97.08%, 98.14%, 94.20%	Not enough performance metrics are measured and did not implement validation metrics.

(*continued on next page*)

Table 1 (*continued*).

Ref.	Year	Method	Contribution	Transaction data	Evaluation metric	Metric value	Limitation
[33]	2018	Multiple Classifier	Credit card fraud detection	Credit card dataset	Accuracy, Precision, Recall	99.90%, 85.85%, 86.70%	Not enough performance metrics are measured and did not tackle the imbalanced data.
[34]	2018	AdaBoost, and Majority Voting	Credit card fraud detection	2013 European bank data [42]	Accuracy, Recall, TNR	99.90%, 78.90%, 99.9%	Not enough performance metrics are measured.
[35]	2018	Conventional Neural Network (CNN)	Fraud detection in credit card transactions	Credit card fraud data (Real-time)	Accuracy	96.5%	Cross validation metrics are not enough.
[36]	2017	Ensemble Learning	Imbalance class address	2013 European bank data [42]	Accuracy, Recall, TNR, AUC	90.60%, 81.80%, 99.5%, 97.76%	Not enough performance metrics are measured and did not tackle the imbalanced data.
[37]	2017	Decision tree (DT) model	Fraud detection in credit card transactions	Single credit card data	NA	NA	Did not focus on performance evaluation.
[38]	2017	Decision tree (DT) model	Credit card default payment detection	Taiwan credit card default payment data	Accuracy, AUC	69.36%, 87.90%	Cross validation is not implemented and did not tackle the imbalanced data.
[39]	2017	Random Forest (RF)	Credit card default payment detection	Taiwan credit card default payment data	Accuracy, AUC, TPR	81.60%, 76.40%, 37.10	Did not tackle the imbalanced data.
[40]	2016	Multilayer Perceptron (MLP), Radial Basis Function (RBF), and Naïve Bayes (NB)	Credit card default payment detection	2005 Taiwan credit card default payment data	Accuracy, Sensitivity, Specificity	77.8%, 53.36%, 88.13%	Cross validation is not implemented and did not tackle the imbalanced data.
[41]	2016	Naïve Bayes (NB) and Random Forest (RF)	Credit card default payment detection	Taiwan credit card default payment data	TPR	81.60%	Not enough performance metrics are measured and did not tackle the imbalanced data.

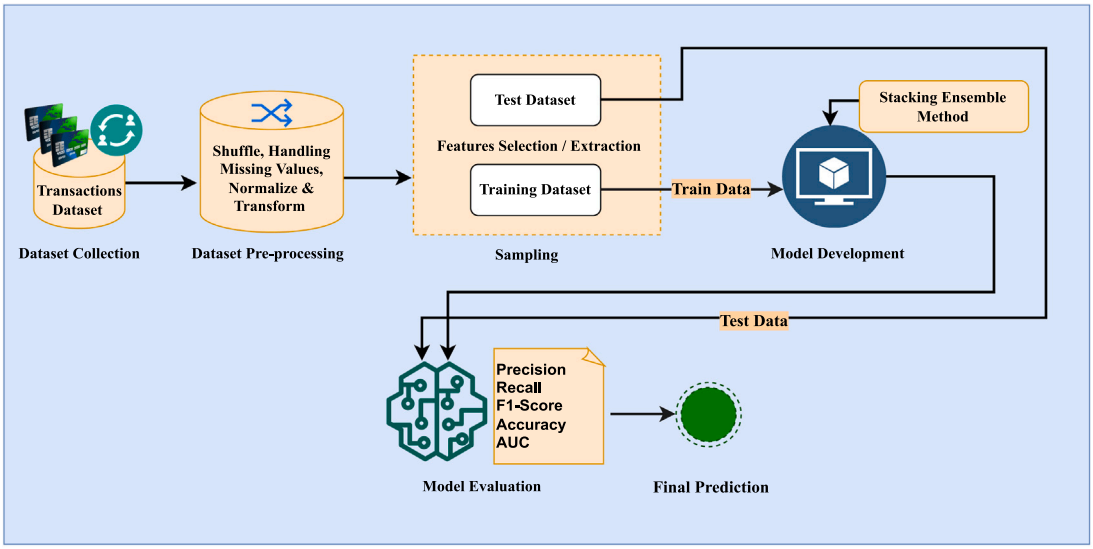


Fig. 1. Block diagram of CCAD architecture.

In this model, we utilize an ensemble learning approach to classify a credit card instance as non-fraud, fraud, non-default payment, or default payment. The overall architecture of credit card transaction anomaly identification is presented in Fig. 1. The proposed model consists of the following steps: (1) Data pre-processing (2) Sampling (3) Model development (apply stacking ensemble method) (4) Model evaluation and anomaly detection.

In the following sections, we describe all steps of the proposed model.

3.1. Data pre-processing and feature selection

Before feeding dataset to the model, we require to examine data integrity and identify missing values. In this study, dataset used to

train our model does not have any missing or null values. We used two publicly available datasets, and a description of these two dataset has been provided in Section 4.1. The number of target classes in both datasets is two. In the *CCF* dataset: One is genuine or non-fraudulent transactions, and the other is fraudulent transactions. In the *CCDP* dataset: One is non-default payment transactions, and the other is default transactions. Fraudulent transactions are 0.1727% on the *CCF* dataset. In contrast, default payments account for 22.12% of the *CCDP* dataset. Given the relatively small proportion of fraudulent transactions in the dataset, to focus on fraud identification is more appropriate.

We require to remove repetitious or duplicate information from the dataset to improve the machine learning model performances. In this work, an XGBoost model is fitted to obtain the features' importance [44]. Upon analyzing the feature importance for model estimation, we observed that features with an importance level below 0.014 do not significantly contribute to the model estimation.

Therefore, in this research, we set the minimum importance level to 0.014 in order to identify the features for further investigation. This resulted in a reduction of the number of features from 29 to 22 in the CCF dataset and from 25 to 22 in the CCDP dataset.

The feature selection algorithm, shown in *Algorithm 1*, involves storing the importance levels of all features in an array called $feat_i$, which are generated using XGBoost. The column numbers and importance values are denoted by key and F_l, respectively. Then, we iterate through each number in $feat_i$ and check if the importance level F_l is greater than 0.014. If the condition is met, we select these features for further analysis and store them in i_{cols}.

Algorithm 1: Features Selection.

Input : Feature importance ($feat_i$) learned by XGBoost [44]
Output: Selected features

```
i_cols[∅], k = 0
foreach F_l ∈ feat_i: do
    if F_l > 0.014 then
        i_cols.append(key)
        k = k + F_l
    end
end
```

In this step, we have plotted a histogram for all features to inspect their behavior, as illustrated in Figs. 2 and 3 for the *CCF* dataset and *CCDP* dataset. The Fig. 2 shows that almost all features follow a *Gaussian* distribution, except *V5, V6, V7*. Despite the abnormal behavior observed in features *V2* and *V3*, the remaining features exhibit a bell-shaped curve, indicating that this dataset is suitable for training and testing machine learning algorithms. As shown in Fig. 3, features *X1* and *X12* to *X17* display abnormal behavior, highlighting the need to apply machine learning algorithms to address the issue of anomalies.

After picking the important features, we need to inspect whether the data has duplicate information. Therefore, we plotted a correlation matrix of important features to detect and get off inputting repetitive data on the proposed models, as illustrated in Figs. 4 and 5. This two figures indicates that almost all features are not correlated, and the data is currently prepared to train a learning model.

Following data pre-processing and feature selection tasks, the dataset has been split into the training set and testing set in the 70:30 ratio. The training data has been further split by utilizing stratified k-fold cross-validation to train the proposed model, which is discussed in Section 4.2. This research includes four semi supervised learning algorithms in the base learner of the model. The following section describes all the algorithms.

3.2. Semi supervised learning algorithms

In this article, a two-level stacked ensemble learning technique has been assessed and implemented for the *CCF* and *CCDP* dataset. The proposed model has two layers, the first layer is called a base learner, and the second layer is called a meta learner. Four semi supervised learner models comprise the base learner: (i) *Elliptic Envelope (eEnvelope)*, (ii) *Isolation Forest (iForest)*, (iii) *Local Outlier Factor (LOF)*, and (iv) *One-Class SVM (OCSVM)*. These learning models are mostly used for fraud or anomaly detection.

We have picked these four models for the first-level learner to detect fraud or anomaly detection from credit card transactions. The *XGBoost* classifier is selected as the second level model also called meta learner. The python programming language utilizes the scikit-learn machine-learning package [45] to analyze data for this study. We describe these models in the following section briefly.

3.2.1. Elliptical envelope (eEnvelope)

The *Elliptical Envelope* model also known as *eEnvelope* model is used for outlier or anomaly detection. This model works better if the data has a *Gaussian* distribution, and this research data also shows a *Gaussian* distribution. The algorithm-*Elliptical Envelope*-produces an imaginary elliptic region around a provided *CCF* dataset and *CCDP* dataset. The ellipse's shape and size are determined using the *fast-minimum covariance determinant (FastMCD)* algorithm and repeatedly measuring the *Mahalanobis* distance before deciding the data covariance matrix [46]. This *Mahalanobis* distance is actually calculated as the number of standard deviations of a data point from the mean. In our research work, two important parameters are used: one is $support\_fraction$ and another one is *contamination*. We set the $support\_fraction$ value as 0.7 in the algorithm by following this: $[n\_sample + n\_features + 1]/2$ and its range is (0, 1). On the other hand, the contamination rate defines the ratio of values that are recognized as outliers. Its rage is (0, 0.5). In our research, we set the contamination value as 0.1 in the algorithm. If data falls inside the envelope, then it is called genuine data; otherwise, it is considered an anomaly or fraud by the algorithm.

3.2.2. Isolation forest (iForest)

The *iForest* is another popular and well-known algorithm used to identify ensemble-based unsupervised outliers from the dataset with high precision and linear time complexity, proposed in [47]. This algorithm uses a *random forest* algorithm (ensemble decision trees) under the hood to identify the outliers in the *CCF* dataset and *CCDP* dataset. The *iForest* algorithm splits the anomalies rather than constructing or profiling regular regions and points by allocating a score to each data point. This performs by adopting the benefit of the fact that anomalies or fraud transactions are the minority data points and values of attributes that are quite distinct from the regular instance. The statistics show that the algorithm works better and can identify inconsistencies effectively when the dataset is a high-level dataset. The dataset *CCF* and *CCDP* are high level data. The method has only *two* variables, the first is the number of binary trees to create, and the second is the sub-sample size. The binary trees are created from the random property of the dataset. Then, travel to every tree in the forest and generate anomaly or inconsistency scores for every data point in every tree [48]. In our research study, the *iForest* algorithm computes an inconsistency score based on the path length needed to isolate a data point in binary trees consisting of all data points. This path length averaged over a forest of random trees measures our decision function and normality. Random partitioning generated noticeably shorter or smaller paths for fraud or anomalies. This forest of random trees collectively produces smaller path lengths for particular samples, which are marked as anomalies or inconsistencies.

3.2.3. Local outlier factor (LOF)

This is a *semi-supervised* and *unsupervised* machine learning algorithm. In 2000, Breunig et al. first introduced the *LOF* algorithm to identify abnormal data points by calculating the local deviation of shared data points concerning its neighbors [49]. This *LOF* is a density-based outlier identification method that discovers outliers by

Fig. 2. Distributions of features on CCF dataset.

computing the local deviation of a given data point, which is appropriate for the outlier identification of an inconsistent distribution dataset. The outlier's appraisal is evaluated based on the density among its neighbor points and every data point. The better possibility it is to be recognized as the outlier when the lower the point's density [48]. For the abnormal data points, the ratio of the *LOF* will be higher because outliers come from low-density regions. For example, if the local outlier factor of the p point is three, it means that the joint density of *p's* neighbors is three times bigger than its neighbor density. This research observed that the normal data point has a *LOF* value of 1 to 1.5, the abnormal data point's *LOF* value is much higher than the normal data point's *LOF* value, and the *LOF* rating denotes a point-wide degree. Therefore, we initially calculated the threshold value based on the user-defined contamination rate. A point is represented as an outlier if its local outlier (*LOF*) rating overcomes the threshold value.

3.2.4. One-class svm (OCSVM)

The most popular *SVM* [50,51] algorithm is used to discover a hyperplane that isolates the two classes of data points. Another semi-supervised variant is the *One-ClassSVM* [52], which attempts to learn a decision boundary that acquires the highest split among the origin and the points [53]. For training, only normal data is required to detect the anomalies from the dataset. This algorithm has only one class of data points, and the job is to estimate a hyperplane that isolates the cluster of data points from the anomalies. It uses the kernel's implicit transformation function $\sigma(\cdot)$ to project the data into a higher dimensional space, then learns a hyperplane or decision boundary that isolates most of the data from the origin. Only a tiny portion of data points are got permits to lie on the other side of the hyperplane or decision boundary, and those data points are recognized as outliers. The *Gaussian* kernel guarantees the presence of a hyperplane or decision boundary [53]. However, we observed that all kernel entrances are positive or zero means all data in the kernel space lies in the same quadrant. As a result, the *Gaussian* kernel nicely fits to negotiate with any arbitrary or random dataset. The function $k(\cdot)$ be defined in Eq. (1). Here, υ is the perpendicular vector of the hyperplane and γ is the bias term.

$$k(y) = \upsilon^T \sigma(y) - \gamma \tag{1}$$

To detect the normal points, the *One-ClassSVM* is use the decision function shown in Eq. (2). This $g(y)$ function always returns a positive value for the normal points and generates a negative value for the other case. The algorithm's output is a binary level determining whether the data point is abnormal or normal.

$$g(y) = sgn(k(y)) \tag{2}$$

3.2.5. XGBoost

In 2016, Chen et al. [54] first proposed the scalable end-to-end tree boosting-based *ensemble* machine learning algorithm called *XGBoost*.

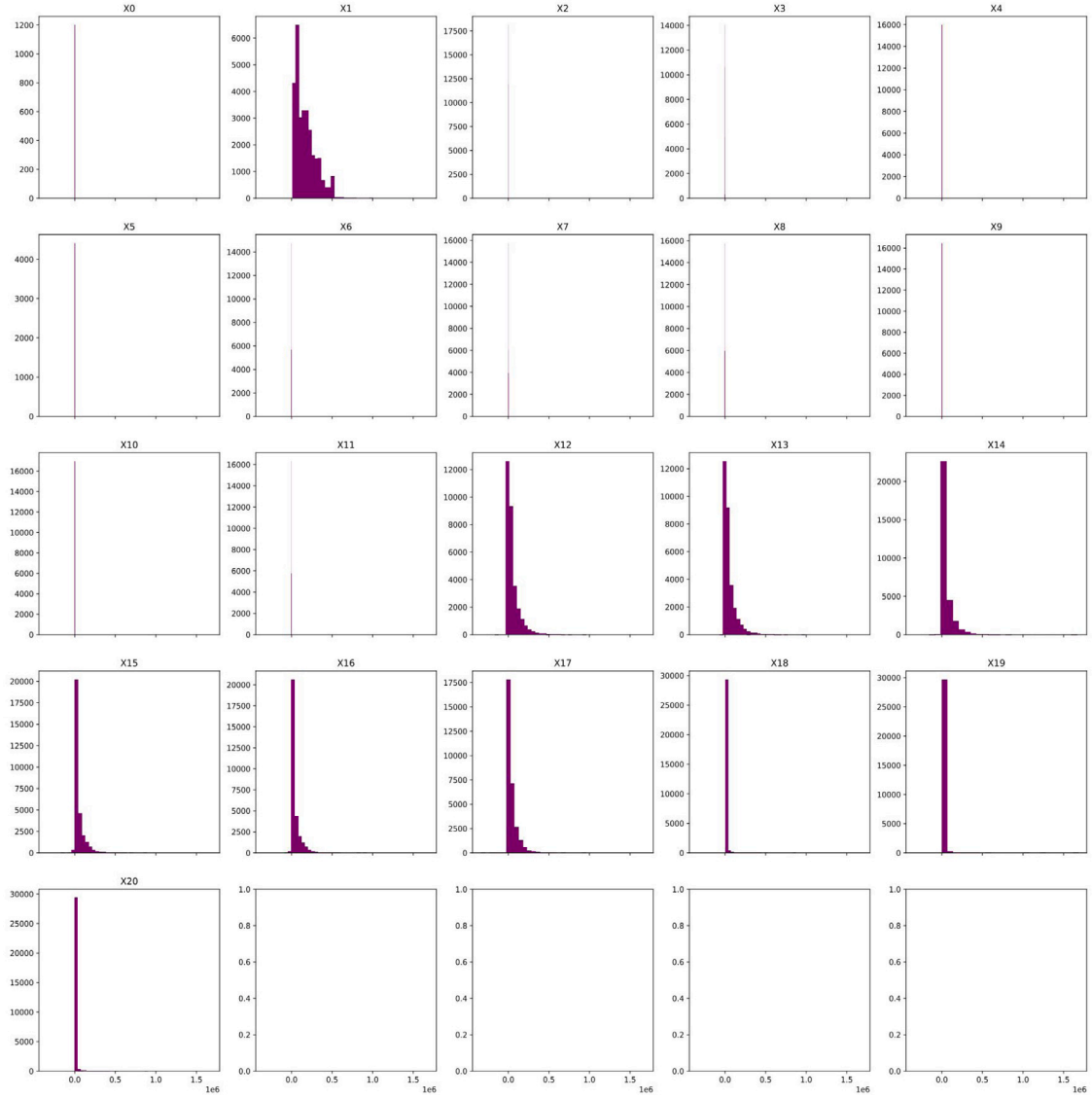
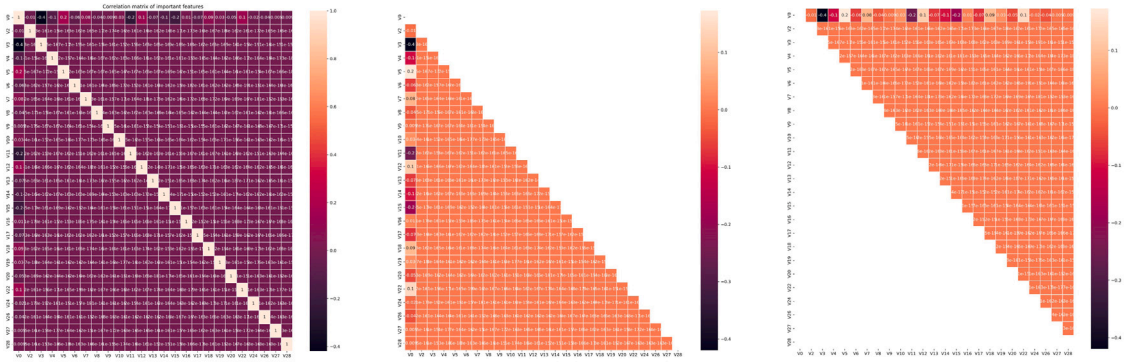


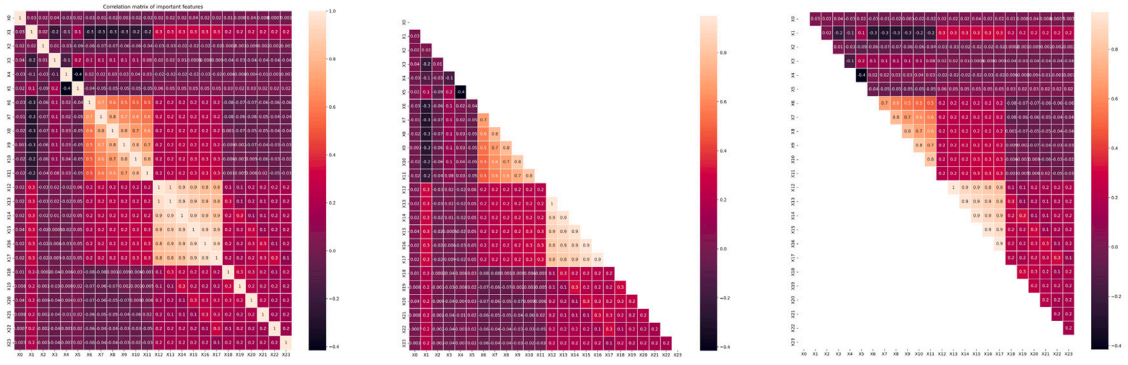
Fig. 3. Distributions of features on CCDP dataset.



(a) Correlation Matrix (b) Correlation Matrix (bottom triangle) (c) Correlation Matrix (top triangle)

Fig. 4. Correlation Matrix of important features on CCF dataset.

This *XGBoost* algorithm is also known as *eXtreme Gradient Boosting*. According to [54], it optimizes the loss function by assembling regularization to address the weighted quantile sketch and sparse data for tree learning. Furthermore, they supply some insights that help to construct a fast and tree-boosting method. These insights hold data sharding, compression, and cache access patterns. This *XGBoost* algorithm beats the other machine-learning algorithms for these tactics and insights in both accuracy and speed. It is also suitable for data engineers or scientists to use the *XGBoost* algorithm in the *GPU* machine or distributed system. Regarding the benefits of this algorithm, we utilized



(a) Correlation Matrix (b) Correlation Matrix (bottom triangle) (c) Correlation Matrix (top triangle)

Fig. 5. Correlation Matrix of important features on CCDP dataset.

this model as a meta learner in our research work. It adopts an add-on learning scheme with 2*nd*-order estimation, 1*st*-order derivative anointed *gradient*, and 2*nd*-order derivative anointed *hessian of loss function*. The *gradient* and *hessian of loss function* are shown in Eqs. (3), (4) and described in the [55].

$$G^{(p)} = \sum_{j=1}^{n} l(y_j, q_j^{(p-1)} + f_p(x_j)) + \mu(f_p) \tag{3}$$

$$G^{(p)} = \sum_{j=1}^{n} \left[g_j f_p(x_j) + \frac{1}{2} h_j (f_p(x_j))^2 \right] + \mu(f_p) \tag{4}$$

Meanwhile, Wang et al. proposed imbalanced *XGBoost* where binary cross entropy is replaced by the focal loss and weighted binary cross entropy to solve the imbalance problem [56], shown in Eqs. (5), (6).

$$L_f = -\sum_{j=1}^{m} y_j (1-\hat{y}_l)^{\beta} log(\hat{y}) + (1-y_j)(\hat{y}_l)^{\beta} log(1-\hat{y}) \tag{5}$$

$$L_{wc} = -\sum_{j=1}^{m} \gamma y_j log(\hat{y}_l) + (1-y_j) log(1-\hat{y}_l) \tag{6}$$

For tuning the class weight, the γ imbalanced parameter is used, while the β parameter manages the shape of the curve. The higher the value of β, the lower the loss and vice-versa.

3.3. Ensemble learning

In the research paper [57], the authors' defined ensemble learning as the mixture of various learning algorithms. It integrates multiple sets of supervised learners to obtain an improved powerful model [33]. For example, two machine learning algorithms will be the same if their contribution or output is averaged out of all possible problems. No single machine learning algorithm can perform other strategies well for all possible data types. For these reasons, it is necessary to combine multiple algorithms. Different kinds of ensemble techniques are available such as *bagging*, *boosting*, and *stacking*. In this research work, we used the *stacking* technique to improve the estimation of our proposal to detect the anomaly or frauds from credit card transactions. The simultaneously used *bagging*, *boosting*, and *stacking* ensemble learning techniques are briefly described below.

Bagging is a parallel ensemble learning method appropriate for complex systems, especially high-variance and low-bias systems. This can reduce the variance but not minimize bias by combining the weak learners. The ensemble technique refers to adding various base estimators to improve one single estimator [58]. Usually, a *bagging* classifier has been utilized to fit the base classifiers on every randomized subset of the dataset, then aggregate each of the particular estimations. A *bagging* classifier can be utilized to reduce the variance of the *BlackBox* classifier, like a decision tree. Reason for introducing randomization in the expansion procedure and expanding the ensemble out of it.

Boosting is a *sequential* ensemble learning approach based on the frame of *PAC* and applies to reduce variance, not bias [59]. Our primary research goal is to identify anomaly in credit card transactions and minimize the classification error rate. *Boosting* technique performs relatively better than the other methods. But there are various basic classifiers and algorithms in the boosting methods, such as *AdaBoost*, *Gradient Boosting Decision Tree*, *XGBoost*, etc. In this research, we used *XGBoost* as a meta-classifier to detect anomaly. The training dataset of this work mainly focuses on the anomaly or fraud identification issue and has the label to identify whether the data is an anomaly or not. Initially, we assigned the weight for each sample in the dataset and basic classifiers to get the probability distribution and then combined them to get a robust classifier.

Stacking is comparable to *boosting* method and is called a *meta-learning* method which produces a symmetric solution to a problem [60]. More formally, *stacking* is an ensemble learning strategy that integrates numerous regression or classification models through metaregression or meta-classifier. In this method, the properties are reformed operating ensemble, and other layers have used these properties. *Stacking* is generally utilized in competitions where various algorithms are hired for data training, and results are averaged. *Overfitting* is a widespread and continually created problem in a model and reduces the functionality of an estimating model in the test set. *Stacking* used the *cross-validation* technique rather than splitting the data into two groups, where datasets are divided into *k-fold* [61] to conquer this problem. In this work, we applied a *stacking ensemble* to develop our proposed model. During the base learner training, the *k-1 fold* is used for training every base learner, and the isolated fold is employed for forecasting. The outputs are eventually the average for all folds. The entire model is made out by *stacking* ensembles which is called *super learner*.

3.4. Proposed credit card anomaly detection model

The ensemble learning technique is one of the most effective machine-learning methods, which is formed by combining a super learner and some base learners. We might achieve better accuracy from this ensemble learner. In our proposed approach, we have used *four* outlier learners as a base learner. We come up with a better *precision* and *accuracy* value by combining several learners. Since fraudulent or default payment transactions are less representative in the dataset, selecting the correct base learner is crucial to identify anomaly transactions. In this article, we have developed a two-layer stacking method for this dataset. The *first layer* has used *four* unsupervised learning models: *eEnvelope*, *iForest*, *LOF*, and *OneClassSVM*. The *XGBoost* as a *meta classifier* is utilized in the *second layer*. All learning algorithms are briefly described in Section 3.2. The proposed architecture is shown in Fig. 6.

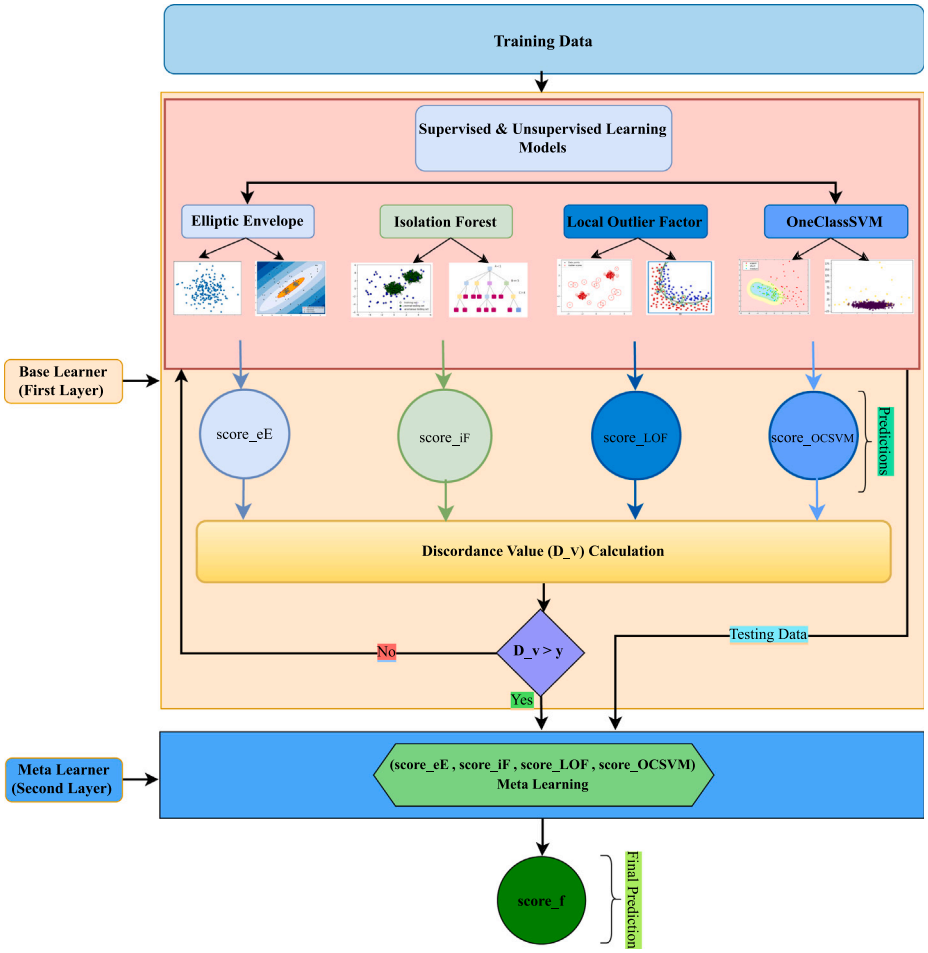


Fig. 6. Proposed stacked ensemble model.

We have also developed a stacked ensemble learning model, as demonstrated in *Algorithm 2*. After completing the pre-processing step on the datasets: *CCF* and *CCDP*, as described in Section 3.1, we used these two datasets D_t as input on the proposed algorithm, and the expected final prediction is $score_f$. Dataset D_t is divided into *five* folds, trained with *four* folds, and the rest of *one* fold is used to testify the model. The test set is verified for every data, and N_d is created for every model, denoting the base layers models outputs as $score_{eE}$, $score_{iF}$, $score_{LOF}$, and $score_{OCSVM}$. Then, we calculated the discordance rate to construct the meta-feature FN_d to fed at the second learning level, as described in Section 4.3.

If $D_v > y$, we keep this estimation value in the final dataset FN_d. Otherwise, go back to step 7 and continue the other transactions procedure. The 0.4 value is set for the y because this value is suitable for this proposed model. FN_d data are fed into the meta-learner or second-level learner, and M_l is input, a stacked ensemble learner. This level generates the final prediction $score_f$ of the credit card anomaly detection.

Algorithm 4 described the overall process of our proposed *CCAD* model. There are three steps: *step 1* is *Preparation of Data*, *step 2* is *Process of Model Training*, and *step 3* is *Model Testing*.

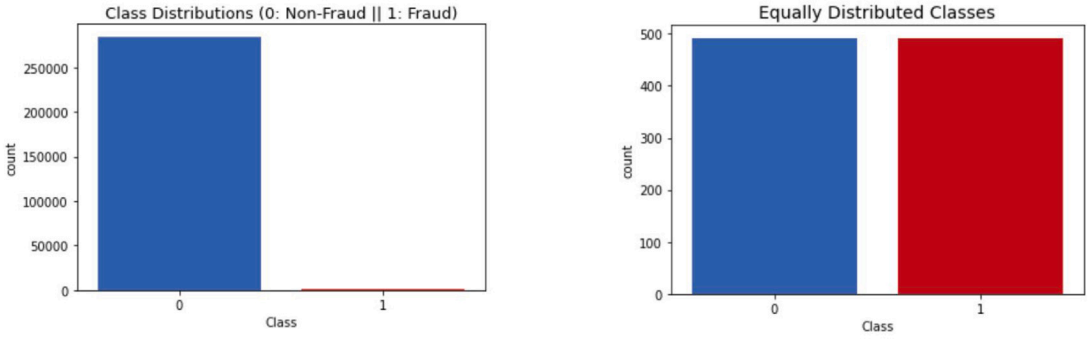
Algorithm 2: Stacked Ensemble Learning.

Input : Dataset D_t after pre-process
Output: Final prediction $score_F$

Step 1 : *learn base − level classifiers*
for $i = 1$ *to* I **do**
  Learn BL_i based on D_t //here, BL_i is Base-label classifier
  $N_d \leftarrow$ *Output of* BL_i
end
Step 2 : *produce new data for metalearner*
for N_d *of each transaction* **do**
  if $D_v > y$ //$y \leftarrow 0.40$ *(Calculated* D_v *using Algorithm* 4*)*
  then
    $FN_d \leftarrow N_d$ //FN_d *is final data for metalearner*
  else
    Go to step 7
  end
end
Step 3 : *learn a meta classifier*
Learn M_l using FN_d data
return $score_f$

4. Experimental settings and *CCAD* framework

This section explains the precision, accuracy, and robustness of our suggested *CCAD* framework and comparison it with the state-of-the-art strategies in this research area. In this study, our prime aim is to enhance the fraud identification effectiveness of the architecture. To acquire our aim, more in-depth acuteness into the data is needed.



(a) Imbalanced Distribution (b) Balanced Distribution

Fig. 7. Tackle imbalanced data on CCF dataset.

Algorithm 3: Proposed *CCAD* Algorithm.

Input : Datasets *CCF* [62] and *CCDP* [40].
Output: Prediction of Anomaly Transactions.
Step 1. Preparation of Data: Dataset is split into the following parts

- Take 70% for training data;
- Keep 30% data for test case;
- Perform Stratified K-Fold Cross Validation on training data;

Step 2. Process of Model Training: Model training is described into following parts

- Tackle *imbalanced* and *over-fitting* issues using stratified K-Fold Cross Validation during training model; *// Using Algorithm 4*
- Trains and validated *eEnvelope*, *iForest*, *LOF*, and *OCSVM* on the entire training data;
- Train a *XGBoost* classifier on the entire training data;
- Calculate *discordance rate* before ensemble the models; *// Using Algorithm 2*
- Train the *meta* classifier using the base learners predictions;

Step 3. Model Testing:

- Apply test data on the proposed model to evaluate the model performances;

4.1. Description of dataset

In this research, we have utilized two imbalanced datasets to solve the problem of imbalanced class distribution and oversampling or overfitting class samples; and enhanced the detection rate of the fraudulent transaction. To the best of our knowledge, general learning algorithms have faced difficulties in tackling those problems and pushed lower identification rates for minority classes.

Dataset one is *Credit Card Fraud (CCF)* dataset is collected from this research works [42,62]. Credit card fraud is a criminal act of unfair profit by making different unauthorized trades using a sufferer's credit card account [42]. As shown in Table 2, the dataset contains 2,84,807 transactions that occurred in two days, and European credit cardholders made these transactions. But it has only 492 fraudulent transactions, which means 0.173% fraud of all transactions and 99.827% non-fraud. The collected dataset was highly unbalanced because the positive class (frauds) accounts for 0.172% of all transactions. It has only numerical input variables, which are the output of the principal components analysis (PCA). The dataset has 31 features (V1 to V28, Time, Amount, Class), but the Time and Amount features are not transformed with PCA, as depicted in Table 3. Due to loyalty matters, the dataset lacks original features and more background information about the data.

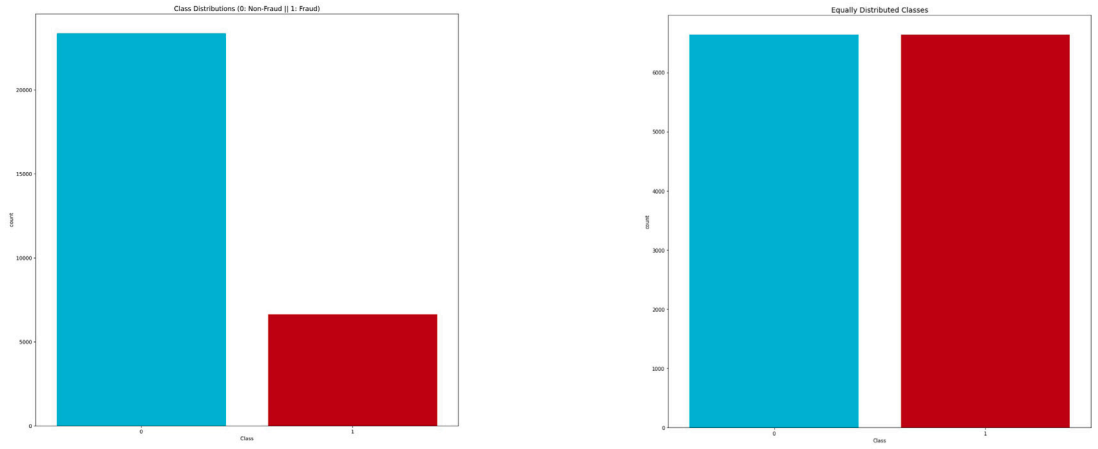
Dataset two is *Credit card default payment (CCDP)* is collected from the research works [40,41,43]. Within the agreed period, if any cardholder fails to repay the minimum amount, it is a default payment [63]. As shown in Table 4, the dataset contains 30,000 payment transactions that occurred from April 2005 to September 2005, and Taiwan bank credit cardholders made these transactions. But it has only 6,636 default payment transactions, which means 22.12% default payment of all transactions and 77.88% non-default payment. The collected dataset was also highly unbalanced because the positive class (default payments) accounts for 22.12% of all transactions. The ratio is almost 1:3 (default payment: non-default payment). The dataset has 25 features (ID, LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE, PAY0, PAY2...PAY6, BILL_AMT1, ...BILL_AMT6, PAY_AMT1, ...PAY_AMT6, and default.payment.next.month), as depicted in Table 5.

Table 6 delivers a quick summary of the formal notation utilized for this research article and described by the authors of [64,65] .

4.2. Data imbalance and overfitting

In the case of binary classification, most of the real-life datasets are highly imbalanced. However, the researchers have recently proposed different approaches, such as [66,67], to handle the imbalanced problem in the dataset. We observe that datasets *CCF* and *CCDP* are imbalanced, and most of the transactions are non-fraud and non-default payments as shown in Figs. 7(a), and 8(a). If we utilize this dataset as a base of our proposed model, this reduces recall and precision. Further, the proposed algorithms suffer from *overfitting problem* because most transactions are *non-fraudulent* or *non-default* transactions regarding two datasets. For this reason, we implement a *stratified* sampling technique to handle an imbalanced dataset, as shown in Fig. 7(b), and apply the *k-fold cross-validation* to address the *overfitting* problem. As we work with a very narrow or biased class, guaranteeing that the number of anomalies is almost the same for all divisions is good. To accomplish this, we apply a *stratified k-fold cross-validation* [61] process that resembles not only k-fold cross-validation but also makes a stratified sample instead of a *random* sample. In *stratified k-fold cross-validation*, the dataset is split so that the data of all classes are allocated evenly in the training set and the test set, depicting the whole dataset in equivalent proportions.

However, after utilizing this technique, the entire dataset is divided into k folds, trained with $k-1$ folds, and then the model is testified by the rest of 1 fold. So, $k = 5$ is the determined value for this research work and goes through 5 left-out folds. Finally, a testset is used to verify all data, and a renewed N_d is created for every model, denoting the base layers models outputs as $score_{eE}$, $score_{iF}$, $score_{LOF}$ and $score_{OCSVM}$ shown in Fig. 6. Then, we calculate the discordance rate to construct the meta-feature FN_d, which is fed at the second learning level.



(a) Imbalanced Distribution (b) Balanced Distribution

Fig. 8. Tackle imbalanced data on CCDP dataset.

Table 2
Details of CCF dataset.

Total transaction $\|L\|$	Legitimate $\|L_+\|$	Fraudulent transactions $\|L_-\|$	Classes	Features
2,84,807	2,84,315	492	2	31

Table 3
CCF dataset attributes.

Attributes	Statement
V1, V2, …,V28	Principal components obtained with PCA
Time	The seconds elapsed between the first transaction and each transaction in the dataset
Amount	Total amount of transaction and can be utilized for dependent cost-sensitive learning
Class	The response variable; Value 1 is for fraud and 0 otherwise

4.3. Discordance value calculation

If the outputs of the respective classifiers are very dissimilar from the data, then the *stacking* technique is particularly relevant. For example, *discordance* value can quickly identify whether all the base-level classifiers have similar estimation outputs or not. If the *discordance* value is high, this means the base classifier estimations are highly dissimilar from each other. For this reason, it is required to know the base-level classifier's outputs and inspect their outcome is almost distinct. Therefore, we have computed the *discordance* rate between the base levels models. This *discordance* value ensures that the ensemble is trained on a distinct *meta* feature instead of being fed with different base-level model predictions, thus guaranteeing not to make a biased model.

The *Algorithm* 4 displays the procedure of discordance value computation. Firstly, we check the four base layer model estimations depicted as $score_{eE}$, $score_{iF}$, $score_{LOF}$, and $score_{OCSVM}$. If the model estimates the fraudulent transaction, the set -1 and otherwise set 1 for the actual transaction. Secondly, sum up four predictions in SoP and check SoP is equal to 4 or -4. If the $SoP = 4$, then all base layers models indicate transactions were genuine, and in the same way, if $SoP = -4$, models agree that transactions are fraudulent. Thirdly, assign value $SSoP = 0$, when the SoP value is 4 or -4; otherwise, set $SSoP = 1$. Finally, we sum up all $SSoP$ values for all the estimation fold transaction data, divide it with the total SoP for that fold, and calculate the discordance value, D_v. This study found that the discordance value is 0.8430232558139535 and 1 for *CCF*, *CCDP*, respectively, which means the first-level model's outputs are almost different. So, *stacking* approach is suitable and applicable to this dataset for the algorithms.

Algorithm 4: Discordance Value (D_v) Computation.

Input : $scores = \{score_{eE}, score_{iF}, score_{LOF}, score_{OCSVM}\}$ *//Set of base layers*
Output: D_v *//Discordance*

$SSoP = 0,\ SoP = 0$
foreach S *in* $scores$ **do**
  if $S == Anomaly$ **then**
    $Set\ S = -1$
  else
    $Set\ S = 1$
  end
  $SoP += S$
  $SSoP += (SoP == 4\ ||\ SoP == -4)\ ?\ 0 : 1$
end
$D_v = SSoP/SoP$
return D_v

4.4. Performance evaluation metrics

The learning algorithm performance measurement showed a transversal behavior in the imbalanced allocation of the classes. So, it is necessary to determine the perfect metrics to evaluate the performance of the classification algorithms and also needs to handle imbalanced datasets. The learning algorithm shows an accuracy paradox for the imbalanced scenario. Moreover, in this study of intrusion detection, accuracy is not the best performance metric; even a model with extreme accuracy might miss-classify the majority of illegal transactions. Therefore, assessing such algorithms for *precision* and *recall* is critical: correctly classifying fraudulent transactions as fraud. In our research, some performance evaluation metrics have been selected: *precision*, *recall*, *F1-score*, *accuracy*, *true positive rate*, *false positive rate*, *mathews correlation coefficients (MCC)*, and *area under the receiver operating characteristic curve (AUC)* [18,19,64,65,68]. Those performance metrics can be measured in the following ways:

Precision is defined as, "From all the fraudulent transactions which model estimated as fraud, number of the fraudulent transaction actually are fraud divided by the total number of frauds estimated by the model

Table 4
Details of CCDP dataset.

Total transaction $\|L\|$	Legitimate $\|L_+\|$	Default-Payment transactions $\|L_-\|$	Classes	Features
30,000	23,364	6,636	2	25

Table 5
CCDP dataset attributes.

Attributes	Statement
ID	Every client id
LIMIT_BAL	Amount of given credit in NT dollars
SEX	Gender (male = 1, female = 2)
EDUCATION	Status of education (graduate school = 1, university = 2, high school = 3, others = 4, unknown = 5, unknown = 6)
MARRIAGE	Status of marital (married = 1, single = 2, others = 3)
AGE	Age in years
PAY0, PAY2, ….PAY6	Status of repayment in September, August, July, June, May, April, 2005, respectively (pay duly = −1, one month delay = 1, ….nine months and above delay = 9)
BILL_AMT1, ….BILL_AMT6	Amount of bill statement in September, August, July, June, May, April, 2005, respectively (NT dollar)
PAY_AMT1, ….PAY_AMT6	Amount of previous payment in September, August, July, June, May, April, 2005, respectively (NT dollar)
default.payment.next.month	Default payment variables (yes = 1, no = 0)

Table 6
Notation utilized for metrics definitions.

Metrics	Symbols
True Positive	T_{11}
False Positive	T_{01}
True Negative	T_{00}
False Negative	T_{10}

as frauds". This metric is employed to identify actual frauds from the fraudulent transactions and calculated by Eq. (7) [19,69].

$$Precision = \frac{T_{11}}{(T_{11} + T_{01})} \tag{7}$$

Recall is defined as, "From all the fraudulent transactions which model estimated as fraud, number of the fraudulent transaction actually are fraud divided by the total number of transactions labeled as fraudulent". This metric helps to answer the question about what proportion of the fraud transaction is detected from fraudulent transactions and measured by Eq. (8) [18,19].

$$Recall = \frac{T_{11}}{(T_{11} + T_{10})} \tag{8}$$

F1-score is defined as, "harmonic average of the recall and precision from the fraudulent transactions dataset, where an F1-score reaches its worst value at 0 and best at 1 (perfect recall and precision) and worst at 0". This metric is used to measure the overall performance of the model and calculated by Eq. (9) [18,19,23].

$$F1 - score = \frac{2 * Recall * precision}{Recall + Precision} \tag{9}$$

Accuracy is defined as, "the ratio of number of fraudulent transactions accurately identified to total number of fraudulent transactions in test dataset". This metric is calculated by using Eq. (10) [18,70].

$$Accuracy = \frac{(T_{11} + T_{00})}{(T_{00} + T_{01} + T_{11} + T_{10})} \tag{10}$$

True positive rate (TPR) is defined as, "From all the fraudulent transactions which model estimated as fraud, number of the fraudulent transaction actually are fraud divided by total number of transactions labeled as fraudulent". TPR, also known as *Recall* and *Sensitivity*. TRP metric helps to answer the question about what proportion of the fraud transaction was detected from fraudulent transactions and measured by Eq. (11) [18,19,64].

$$TPR = \frac{T_{11}}{(T_{11} + T_{10})} \tag{11}$$

False positive rate (FPR) is defined as, "From all the fraudulent transactions, counts the number of transactions is wrongly identified". FRP metric is also known as Type-I error and measured by Eq. (12) [18,19,23].

$$FPR = \frac{T_{01}}{(T_{01} + T_{00})} \tag{12}$$

Area under the ROC curve (AUC) is defined as, "From subsets of the fraudulent and non-fraud or legitimate transactions, Where α exhibit the possibility of all comparisons among these two subsets (i.e., $|L_+|\&|L_-|$). It lies within the interval [0;1], and outputs will be provided by averaging all comparisons, where 1 indicates the best performance" [71,72]. This metric is calculated by Eq. (13). In another way, we can say that the TPR and FPR are plotted on the y-axis and x-axis in the ROC space, respectively. The entire two-dimensional area underneath the ROC curve is calculated by AUC and can be gained by measuring the collaborative performance of all possible classification thresholds [18,25].

$$AUC = \frac{1}{|l_+| * |l_-|} \sum_{1}^{|l_+|} \sum_{1}^{|l_-|} \alpha(l_+, l_-);\ Where, \alpha(l_+, l_-) = \begin{cases} 1, & \text{for } l_+ > l_- \\ 0.5, & \text{for } l_+ = l_- \\ 0, & \text{for } l_+ < l_- \end{cases} \tag{13}$$

Matthews correlation coefficient (MCC) MCC calculates the degree of association between the estimation and genuine class labels. It takes 1 if and only if the estimated and genuine class labels are the same. Otherwise, it takes −1 when the estimated class label and the genuine class complement entirely to each other. Its calculated by using Eq. (14) [68].

$$MCC = \frac{(T_{11} * T_{00}) - (T_{01} * T_{10})}{\sqrt{(T_{11} + T_{01})(T_{11} + T_{00})(T_{00} + T_{01})(T_{00} + T_{10})}} \tag{14}$$

Detection rate is defined as, "how correctly the model is estimating the true positive cases or how precisely the model is tackling the

Table 7
Performance metrics of various individual classifiers on CCF dataset.

Classifiers	Accuracy			Precision			Recall			F1 score			AUC		
	Fraud	*Non-fraud*	*Overall*	*Fraud*	*Non-fraud*	*Overall*	*Fraud*	*Non-fraud*	*Overall*	*Fraud*	*Non-fraud*	*Overall*	*Fraud*	*Non-fraud*	*Overall*
LR	73.13%	96.06%	95.11%	73.21%	96.88%	95.16%	71.07%	93.75%	95.06%	70.30%	92.08%	95.11%	73.30%	96.08%	95.11%
RF	63.23%	96.57%	93.60%	61.35%	97.55%	96.47%	57.70%	99.13%	98.52%	62.30%	96.35%	93.60%	70.30%	96.80%	93.60%
DT	84.25%	95.80%	92.59%	83.25%	97.17%	95.64%	82.51%	98.17%	89.24%	83.21%	97.31%	92.33%	85.20%	95.05%	92.59%
KNN	67.23%	96.31%	94.17%	51.37%	89.23%	92.32%	61.23%	96.25%	96.17%	61.47%	95.47%	94.17%	68.50%	95.08%	95.16%
SVM	80.21%	95.36%	94.59%	81.17%	97.35%	98.43%	77.31%	94.47%	86.11%	81.45%	97.68%	93.31%	80.30%	95.45%	97.64%
OneClassSVM	84.45%	97.31%	98.47%	85.51%	99.09%	99.16%	75.54%	93.60%	94.21%	82.11%	98.24%	97.59%	84.50%	97.61%	98.47%
eEnvelope	93.14%	96.17%	95.73%	91.51%	98.09%	97.60%	92.47%	96.15%	94.87%	92.18%	97.16%	95.61%	93.16%	96.03%	96.30%
iForest	96.34%	98.70%	98.65%	96.32%	98.09%	98.20%	97.31%	98.47%	98.64%	95.06%	98.21%	98.52%	96.60%	98.08%	98.75%
LOF	97.45%	99.36%	98.70%	96.32%	98.62%	99.09%	96.25%	96.03%	96.11%	97.31%	98.80%	98.25%	97.71%	98.80%	98.89%
XGBoost	86.34%	98.50%	98.09%	90.32%	98.09%	98.94%	92.31%	98.57%	97.23%	93.06%	98.51%	98.08%	86.34%	98.60%	98.09%
GBC	67.56%	83.36%	85.73%	72.32%	88.62%	86.74%	76.25%	86.03%	84.35%	77.31%	86.80%	85.53%	67.57%	83.36%	85.73%

Table 8
Performance metrics of various individual classifiers on CCDP dataset.

Classifiers	Accuracy			Precision			Recall			F1 score			AUC		
	Default	*Non-default*	*Overall*	*Default*	*Non-default*	*Overall*	*Default*	*Non-default*	*Overall*	*Default*	*Non-default*	*Overall*	*Default*	*Non-default*	*Overall*
LR	54.09%	75.35%	79.19%	47.32%	78.15%	76.25%	45.14%	87.48%	86.75%	51.21%	74.56%	78.90%	54.23%	75.60%	79.39%
RF	67.16%	81.60%	80.70%	72.25%	82.15%	81.09%	58.14%	84.36%	83.30%	68.16%	82.35%	84.25%	67.36%	81.57%	80.70%
DT	61.78%	75.36%	75.69%	57.58%	82.34%	78.90%	55.16%	85.19%	82.27%	62.35%	78.69%	79.26%	60.47%	75.66%	75.29%
KNN	48.14%	77.25%	76.89%	43.35%	80.21%	77.86%	45.25%	84.69%	81.35%	49.16%	78.16%	77.35%	48.25%	77.36%	76.92%
SVM	64.35%	80.32%	78.33%	67.45%	81.18%	79.32%	65.16%	82.14%	77.62%	65.19%	84.09%	80.21%	64.16%	81.11%	78.75%
OneClassSVM	79.25%	92.06%	91.11%	73.23%	90.16%	90.14%	78.25%	86.17%	89.72%	80.09%	92.31%	91.75%	79.00%	92.15%	91.15%
eEnvelope	63.16%	85.14%	82.78%	62.09%	86.11%	83.39%	64.77%	87.24%	89.09%	63.70%	85.50%	82.90%	63.25%	85.30%	82.80%
iForest	66.22%	90.05%	88.15%	64.10%	91.04%	90.12%	66.21%	91.17%	89.08%	66.70%	91.10%	89.21%	66.25%	91.08%	88.17%
LOF	76.80%	87.25%	85.59%	74.48%	88.64%	86.47%	70.15%	91.00%	84.80%	77.09%	88.55%	85.90%	77.11%	87.90%	86.00%
XGBoost	73.34%	88.50%	91.09%	70.32%	88.09	90.94%	72.31%	85.57%	89.23%	73.06%	89.51%	91.18%	73.34%	88.60%	92.14%
GBC	63.16%	79.06%	81.33%	71.22%	84.12%	81.24%	72.25%	80.03%	80.35%	67.31%	80.08%	81.53%	67.57%	82.25%	82.93%

fraudulent cases in the fraudulent transactions". This metrics can be measured by Eq. (15) [24].

$$Detection\ rate = \frac{T_{11}}{(T_{11} + T_{10})} \tag{15}$$

4.5. Experimentation results and discussion

This section presents the simulation results of our proposed model CCAD and comparisons between existing works. In Section 4.5.1, we discuss the experimental results performed for the single classifiers, and then we discuss the performance results of the proposed ensemble model in Section 4.5.2. In Section 4.5.3, we compare the results between single classifiers and the proposed framework. Also, we compare and present the framework results with existing works.

4.5.1. Results discussion of single classifiers

In our study, we first use several classifiers to identify the *accuracy*, *precision*, *recall*, *F1-score*, and *AUC* value based on frauds and non-frauds according to the *CCF* dataset, as shown in Table 7. The *precision* value is relatively high, and the average is 96.159%. For the fraud class, the *precision* value changes from 51% to 96% with respect to *KNN* and *LOF* classifiers, but the non-fraud class *precision* values are pretty similar, which means non-fraud class *precision* values are dominant to the overall *precision* results.

The same procedure has been followed for the *CCDP* dataset to identify the *accuracy*, *precision*, *recall*, *F1-score*, and *AUC* values based on *default* and *non-default* payments as shown in Table 8. The *precision* value is relatively not so high, and its average value is 64.00%. For the default-payment class, the *precision* value is changed from 47% to 73%, but for the non-default payment class, *precision* values are pretty similar, which means non-default payment class *precision* values dominate the overall *precision* results.

After analyzing all outcomes of the several classifiers, we found that *single classifiers* are weaker to classify from the dataset when it includes *imbalanced* distribution, but at the same time, ensemble multiple classifiers provide us better outcomes. From this motivation, we propose a new *stacked ensemble learning* framework that performs better than other ensemble learning approaches. To create a new framework, we first choose the best five single classifiers models, and they are *SVM*, *OneClassSVM*, *eEnvelope*, *iForest*, and *LOF*, with their overall predicted results from the *CCF* dataset and *CCDP* dataset, as shown in Table 9, Table 10 in term of *accuracy*, *precision*, *recall*, *f1-score*, and *AUC*.

Consequently, the best five single classifiers' performance over *CCF* dataset and *CCDP* dataset as shown in Fig. 9(a), Fig. 9(b), respectively. In our proposed framework, the first layer is the base learner. We selected four learners from this Table 9 as a base learner in our framework. However, we did not choose the *SVM* classifier as a learner in our model because the *OneClassSVM* classifier provides us better outcomes when we ensemble with other single classifiers.

Our analysis shows that for the case of *CCF* dataset, the *OneClassSVM* and *iForest* classifiers performs better for all-most all evaluation criteria compared to others classifiers except the recall value of *OneClassSVM*. The *OneClassSVM* classifiers outcomes are 98.47% *accuracy*, 99.16% *precision*, 97.59% *F1-score*, and 98.47% *AUC* and *iForest* classifiers achieved 98.65% *accuracy*, 98.20% *precision*, 98.64% *recall*, 98.52% *F1-score*, and 98.75% *AUC*. Moreover, *LOF* classifier achieved 98.70% *accuracy*, 99.09% *precision*, 96.11% *recall*, 98.25% *F1-score*, and 98.89% *AUC*. The *eEnvelope* classifier outcomes are less than other classifiers' outcomes, but it performs better when we ensemble it with different classifiers in our model.

For the case of *CCDP* dataset, the *OneClassSVM* classifier demonstrates excellent outcome for all-most all evaluation criteria compared to other classifiers except the *recall* value. This classifier gained an *accuracy* of 91.11%, a *precision* of 90.14%, a *recall* of 89.72%, a *F1-score* value of 91.75%, a *AUC* value of 91.15% and a *specificity* of 92.13%. The *eEnvelope* classifier performance results are less than other classifiers' outcomes, but it performs better when we ensemble it with different classifiers in our proposed model.

4.5.2. Results discussion of proposed ensemble method

In the previous Section 4.5.1, we already have discussed that we choose the best four single classifier models from the nine models to utilize in our proposed framework as a base learner. This section describes the results performance of the first layer or base layer and then the overall performance of the stacked ensemble model.

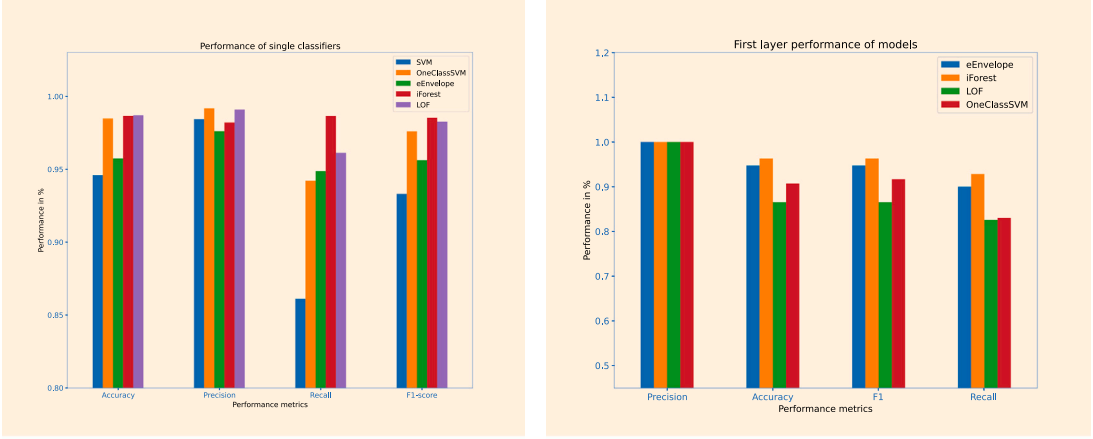
Upon computing the experiment on *CCF* dataset, the performance of the first layer or base learner of the model is promising, and we found that the performance of precision is 0.9992, 0.9995, 0.9984, and 0.9988 for eEnvelope, iForest, LOF, and OneClassSVM, respectively. But the iForest classifier achieves better accuracy compared to

Table 9
Overall predicted results of OneClassSVM, eEnvelope, iForest, and LOF on CCF dataset.

Name of classifiers	Performance evaluation					
	Accuracy	*Precision*	*Recall*	*F1-score*	*AUC*	*Specificity*
SVM	94.59%	98.43%	86.11%	93.11%	97.64%	94.78%
OneClassSVM	98.47%	99.16%	94.21%	97.59%	98.47%	98.62%
eEnvelope	95.73%	97.60%	94.87%	95.61%	96.30%	98.77%
iForest	98.65%	98.20%	98.64%	98.52%	98.75%	98.61%
LOF	98.70%	99.09%	96.11%	98.25%	98.89%	98.73%

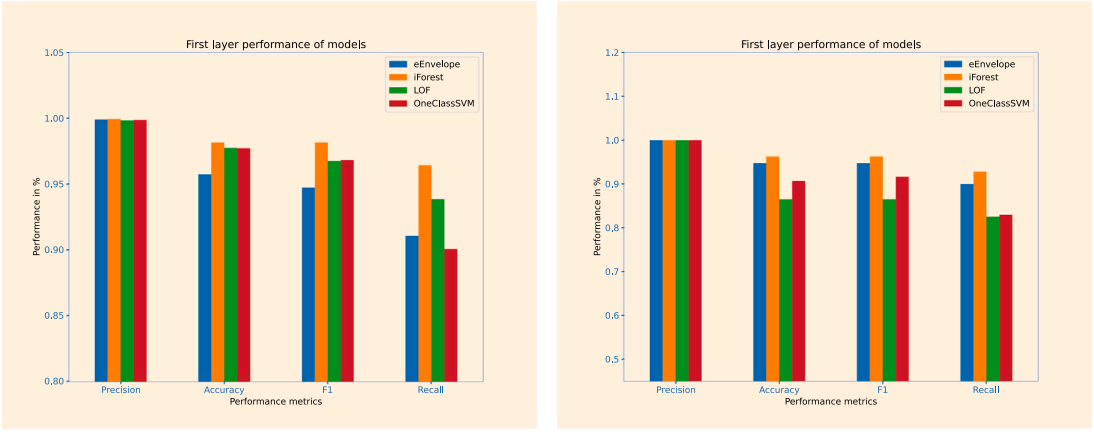
Table 10
Overall predicted results of OneClassSVM, eEnvelope, iForest, and LOF on CCDP.

Name of classifiers	Performance evaluation					
	Accuracy	*Precision*	*Recall*	*F1-score*	*AUC*	*Specificity*
SVM	78.33%	79.32%	77.62%	80.21%	78.75%	77.83%
OneClassSVM	91.11%	90.14%	89.72%	91.75%	91.15%	92.13%
eEnvelope	82.78%	83.39%	89.09%	82.90%	82.80%	81.57%
iForest	88.15%	90.12%	89.08%	89.21%	88.17%	89.06%
LOF	85.59%	86.47%	84.80%	85.90%	86.00%	84.91%



(a) CCF dataset (b) CCDP dataset

Fig. 9. Performances of the single classifiers on CCF & CCDP dataset.

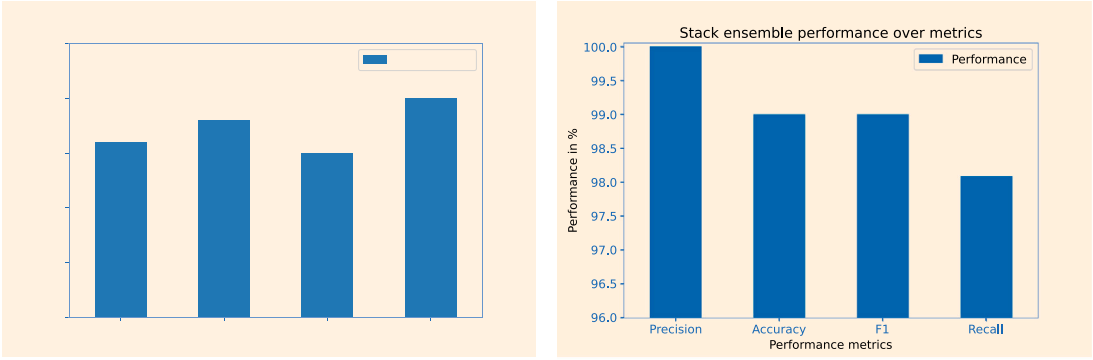


(a) CCF dataset (b) CCDP dataset

Fig. 10. First layer performance of proposed CCAD model on CCF & CCDP dataset.

other classifiers. The first layer performance of the model is shown in Fig. 10(a).

In the experiment on the *CCDP* dataset, the base learner achieved a precision of 1, 1, 1, and 1 for eEnvelope, iForest, LOF, and OneClassSVM, respectively. Moreover, iForset classifier gains better accuracy, recall, and F1-score results than other classifiers. The iForset classifier obtains an accuracy of 0.9627, recall of 0.9280, and F1-score of 0.9627 as shown in Fig. 10(b).



(a) CCF dataset (b) CCDP dataset

Fig. 11. Performance of the proposed CCAD model on CCF & CCDP dataset.

Every newish column will have been built using only the test folds after the *K-Fold Cross-Validation* process ends, as explained in Section 4.2. Then *accuracy*, *precision*, *recall*, and *f1-score* are measured for every newish column created for the base classifiers. Finally, the proposed framework performance has been measured and contrasted with the last layer outputs with the actual labels on *CCF* dataset, as shown in Fig. 11(a). From Fig. 11(a), we see that the proposed framework gained 99.96% *precision* or *PPV*, 100% *recall* or *TPR*, 99.95% *F1-score*, 99.98% *accuracy* and it stunning nicely performative for fraud identification, getting almost 100% score for all the performance metrics.

The same procedure has been followed for the *CCDP* dataset, and stacked ensemble performance has been shown in Fig. 11(b). The proposed model achieves a precision value of 100%, a recall of 98.09%, a F1-score of 98.83%, and an accuracy of 99%. The results show that the proposed model performs better in detecting default payments or anomalies.

Moreover, we have used another appropriate metric called the *ROC* analysis with *AUC* to evaluate the proposed model performance with the *imbalance* dataset due to the vulnerability of classifiers to imbalanced datasets. Firstly, we generate the *ROC* curve concerning the *AUC* metric regarding the anomalous data to ensure the proposed approach can split the *non-fraud* and *fraud* transactions. Then we plot the true positive rate (*TRP*) against the false positive rate (*FPR*) to create the receiver operating characteristics curve, and the graph under the *ROC* curve is measured as the *AUC*. The *ROC* curve illustration is shown in Fig. 12(a). This *ROC* curve show that the presented approach gains a promising *AUC* of 98% on *CCF* dataset for both the *non-fraud* and *fraud* classes. For the *CCDP* dataset, the *ROC* curve illustration is shown in Fig. 12(b), and the proposed model achieves a better *AUC* of 99% for both the *non-default* payment and *default* payment classes.

4.5.3. Comparative assessment

In this section, we have depicted some comparisons of the proposed framework results among the existing frameworks, which they use to classify the *CCF* dataset and *CCDP* dataset using a single classifier.

From Table 11, we see that the proposed model achieves 97.86% of *precision* value for the fraud classes, which is better than the other single classifiers' *precision* values. It improves the *precision* values for the fraud classes by about 14.44%, 6.94%, 1.60%, and 1.39% than the best single classifiers precision gaining models of *OneClassSVM*, *eEnvelope*, *iForest*, and *LOF*, respectively. Compared to other existing works, this model can be better performed and enhanced in the fraudulent cases detection from the *CCF* dataset, which is, on average 10.41% and also shows better performance in the non-fraud class (99.99%). However, other researchers' works are also suitable for real transaction detection. Moreover, the presented model can achieve a better *precision* value, which is 99.96% in the overall cases.

For the case of *CCDP* dataset, the proposed model gains a precision of 93.53% on default payment classes, which is better than the other single classifiers *precision* values. It enhances the *precision* values for the *default* payments by about 38.67%, 27.72%, 50.63%, 45.91%, and 25.58% than the best single classifiers precision gaining models of SVM, OneClassSVM, eEnvelope, iForest, and LOF, respectively. Compared to other existing works, this presented model can be better performed and enhanced in the default payment cases detection from the *CCDP* dataset, which is, on average 41.82% and also shows better performance in the non-default payment class (99.52%). Moreover, the model can achieve a better *precision* value, which is 100% in the overall cases (see Table 12).

After comparing the proposed works with the single classifiers, we compare our works' performance with other existing works in terms of the ROC curve (AUC). From Tables 13 and 14, we clearly see that our model obtains an *AUC* of 98.94% from the *CCF* dataset and 99.00% from the *CCDP* dataset, which is the best regarding the other works, and it ultimately outperforms existing works technique to detect anomalies.

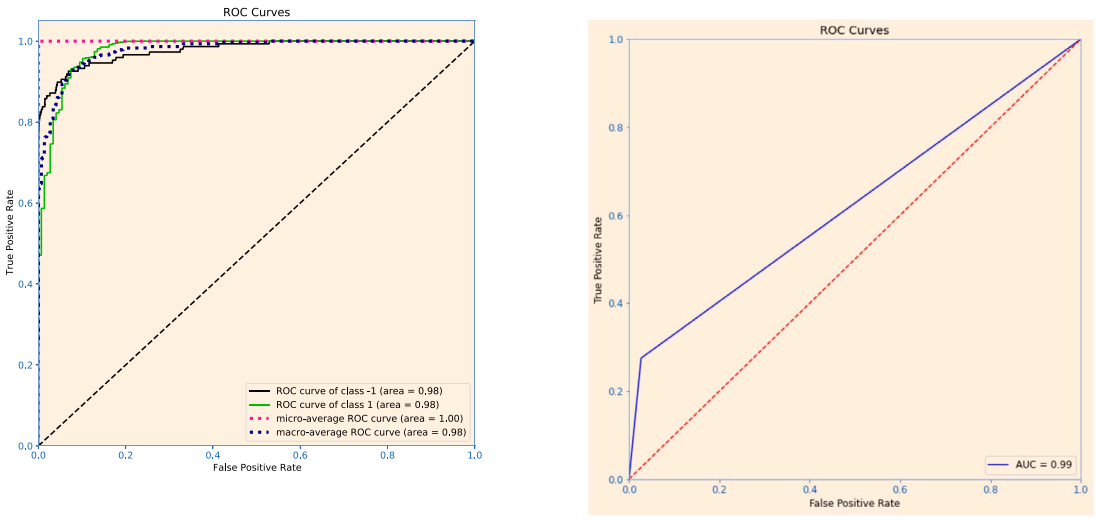
As shown in Table 15, the proposed model achieves the best precision of 99.96% in the overall cases. Moreover, it gains 97.86% precision from fraud cases and 99.99% precision from non-fraud cases. Compared to other existing works, this proposed model can enhance the performance to detect fraud from the CCF dataset.

From Table 16, we see that the proposed model achieves 100% of precision from the overall cases, 93.53% of precision from default cases, and 99.52% precision from non-default cases. Compared to other existing works, this proposed model can enhance the performance to detect anomalies from the CCDP dataset.

Tables 17, 18 shows the overall comparison between the presented approach and existing state-of-the-art using ensemble approaches in handling the imbalanced class distribution on the *CCF* dataset and *CCDP* datasets, respectively.

From the *CCF* dataset, we exceed by acquiring the *maximum TPR* for the minority class, which is 0.96. The presented proposed approach also gains an *accuracy* of 99.98%, and provides an excellent *TPR* or *recall* of 100%, a *precision* of 99.96%, an *f1-score* of 99.95%, which is better than the state-of-the-art approach. Thus, considering all performance metrics, the proposed technique outperforms the other state-of-the-art fraud detection from the *CCF* dataset. The proposed *CCAD* model can show higher performances by tackling the unbalanced class distribution problem of the *CCF* dataset.

From the *CCDP* dataset, the proposed model gains *maximum TPR* of 0.98 for the minority class, an *accuracy* of 99.00%, an excellent *TPR* or *recall* of 98.09%, a *precision* of 100%, an *f1-score* of 99.00%, which is compared to better than other state-of-the-art approaches. After considering all performance metrics, the proposed ensemble *CCAD* model beats the other state-of-the-art anomaly detection from the *CCDP* dataset. This model can show higher performances by tackling the unbalanced class distribution problem of the *CCDP* dataset.



(a) ROC curves on CCF dataset (b) ROC curves on CCDP dataset

Fig. 12. ROC curves of proposed model on CCF & CCDP dataset.

Table 11
Comparison between the proposed model and single classifiers performances in terms of accuracy, precision, recall, F1-score, and AUC on *CCF* dataset.

Classifiers	Accuracy			Precision			Recall			F1-score			AUC		
	Fraud	*Non-fraud*	*Overall*	*Fraud*	*Non-fraud*	*Overall*	*Fraud*	*Non-fraud*	*Overall*	*Fraud*	*Non-fraud*	*Overall*	*Fraud*	*Non-fraud*	*Overall*
DT	84.25%	95.80%	92.59%	83.25%	97.17%	95.64%	82.51%	98.17%	89.24%	83.21%	97.31%	92.33%	85.20%	95.05%	92.59%
KNN	67.23%	96.31%	94.17%	51.37%	89.23%	92.32%	61.23%	96.25%	96.17%	61.47%	95.47%	94.17%	68.50%	95.08%	95.16%
SVM	80.21%	95.36%	94.59%	81.17%	97.35%	98.43%	77.31%	94.47%	86.11%	81.45%	97.68%	93.31%	80.30%	95.45%	97.64%
OneClassSVM	84.45%	97.31%	98.47%	85.51%	99.09%	99.16%	75.54%	93.60%	94.21%	82.11%	98.24%	97.59%	84.50%	97.61%	98.47%
eEnvelope	93.14%	96.17%	95.73%	91.51%	98.09%	97.60%	92.47%	96.15%	94.87%	92.18%	97.16%	95.61%	93.16%	96.03%	96.30%
iForest	96.34%	98.70%	98.65%	96.32%	98.09%	98.20%	97.31%	98.47%	98.64%	95.06%	98.21%	98.52%	96.60%	98.08%	98.75%
LOF	97.45%	99.36%	98.70%	96.52%	98.62%	99.09%	96.25%	96.03%	96.11%	97.31%	98.80%	98.25%	97.71%	98.80%	98.89%
Proposed Approach	**98.83%**	**99.99%**	**99.98%**	**97.86%**	**99.99%**	**99.96%**	**99.83%**	**99.99%**	**100.00%**	**99.87%**	**99.93%**	**99.95%**	**98.73%**	**99.86%**	**98.94%**

Table 12
Comparison between the proposed model and single classifiers performances in terms of accuracy, precision, recall, F1-score, and AUC on *CCDP* dataset.

Classifiers	Accuracy			Precision			Recall			F1-score			AUC		
	Default	*Non-default*	*Overall*	*Default*	*Non-default*	*Overall*	*Default*	*Non-default*	*Overall*	*Default*	*Non-default*	*Overall*	*Default*	*Non-default*	*Overall*
DT	61.78%	75.36%	75.69%	57.58%	82.34%	78.90%	55.16%	85.19%	82.27%	62.35%	78.69%	79.26%	60.47%	75.66%	75.29%
KNN	48.14%	77.25%	76.89%	43.35%	80.21%	77.86%	45.25%	84.69%	81.35%	49.16%	78.16%	77.35%	48.25%	77.36%	76.92%
SVM	64.35%	80.32%	78.33%	67.45%	81.18%	79.32%	65.16%	82.14%	77.62%	65.19%	84.09%	80.21%	64.16%	81.11%	78.75%
OneClassSVM	79.25%	92.06%	91.11%	73.23%	90.16%	90.14%	78.25%	86.17%	89.72%	80.09%	92.31%	91.75%	79.00%	92.15%	91.15%
eEnvelope	63.16%	85.14%	82.78%	62.09%	86.11%	83.39%	64.77%	87.24%	89.09%	63.70%	85.50%	82.90%	63.25%	85.30%	82.80%
iForest	66.22%	90.05%	88.15%	64.10%	91.04%	90.12%	66.21%	91.17%	89.08%	66.70%	91.10%	89.21%	66.25%	91.08%	88.17%
LOF	76.80%	87.25%	85.59%	74.48%	88.64%	86.47%	70.15%	91.00%	84.80%	77.09%	88.55%	85.90%	77.11%	87.90%	86.00%
Proposed Approach	**93.64%**	**99.63%**	**99.00%**	**93.53%**	**99.52%**	**100%**	**96.21%**	**98.00%**	**98.09%**	**93.35%**	**99.06%**	**98.83%**	**93.21%**	**99.72%**	**99.00%**

Table 13
Performance of AUC comparison between existing works and proposed approach on *CCF*.

Ref.	Approach	Base learners	Performance of AUC
[18]	Boosting (CatBoost)	KNN + CatBoost	96.94%
[24]	Bagging + Boosting (AdaBoost)	AdaBoost + RandomForest	97.08%
[25]	Ensemble Learning	RNN + FFNN	83.37%
[29]	Ensemble Learning	KNN + DT + MLP	97.40%
[31]	Multiple Classifier	C4.5 + NB	87.90%
[36]	Ensemble Learning	Deep Belief Network	97.76%
Proposed Approach	**Multiple Classifier + Ensemble Learning**	**eEnvelope + iForest + LOF + OneClassSVM**	**98.94%**

4.5.4. Complexity analysis

This section briefly presented the proposed *CCAD* model complexity. In our research work, we have used well-known outlier detection algorithms to identify the anomalies in credit card transactions. Proposed *Algorithm 3* has three steps and in every step we calculated the complexity: $O(s) + O(k \cdot n)$, $O(4 \cdot n \cdot t) + O(n) + O(p \cdot q)$, and $O(r)$. Table 19 summarizes the time complexity. The complexity of the proposed model is presented in Table 19.

5. Conclusion

The electronic credit card serves as an alternative to cash payments. However, some cardholders may not honor their repayment obligations or may misuse their cards. In addition, transactions can be deemed fraudulent when unauthorized individuals or groups illicitly use these electronic credit cards. In the credit card dataset, features like imbalanced class distribution and overlapping class issues have posed challenges for researchers. We proposed the *CCAD* model to solve the issues as discussed above, and also detection rate of anomalies is

Table 14
Performance of AUC comparison between existing works and proposed approach on *CCDP* dataset.

Ref.	Approach	Base learners	Performance of AUC
[19]	Boosting (AdaBoost, LGBM)	AdaBoost + LGBM	82.00%
[23]	Ensemble Learning	BPNN + SVM + RF	97.10%
[27]	Multiple Classifier	ANN + SVM	93.70%
[28]	Multiple Classifier	NB + KNN + LR	91.00%
[31]	Multiple Classifier	C4.5 + NB	70.30%
[38]	Boosting (XGBoost)	CART	87.90%
[39]	Bagging	RF	76.40%
Proposed Approach	**Multiple Classifier + Ensemble Learning**	**eEnvelope + iForest + LOF + OneClassSVM**	**99.00%**

Table 15
Overall precision comparison between existing works and proposed approach on *CCF* dataset.

Ref.	Approach	Base learners	Precision evaluation		
			Fraud	*Non-Fraud*	*Overall*
[21]	Boosting (AdaBoost)	AdaBoost + OCSVM	79.50%	95.00%	97.00%
[22]	Ensemble Learning	iForest + RandomForest	85.80%	94.38%	98.30%
[25]	Ensemble Learning	RNN + FFNN	80.73%	99.20%	95.69%
[26]	Multiple Classifier	ANN + KNN + SVM	80.56%	98.35%	97.43%
[30]	Multiple Classifier	KNN + DT	75.40%	91.30%	85.11%
[31]	Multiple Classifier	C4.5 + NB	74.32%	88.53%	86.45%
[32]	Multiple Classifier	RandomForest	79.80%	96.48%	98.14%
[33]	Multiple Classifier	RF + FFNN	56.20%	98.70%	85.85%
Proposed Approach	**Multiple Classifier + Ensemble Learning**	**eEnvelope + iForest + LOF + OneClassSVM**	**97.86%**	**99.99%**	**99.96%**

Table 16
Overall precision comparison between existing works and proposed approach on *CCDP* dataset.

Ref.	Approach	Base learners	Precision evaluation		
			Default	*Non-default*	*Overall*
[19]	Boosting (AdaBoost, LGBM)	AdaBoost + LGBM	82.00%	99.00%	97.00%
[28]	Multiple Classifier	NB + KNN + LR	78.38%	97.56%	97.00%
[31]	Multiple Classifier	C4.5 + NB	71.95%	89.93%	83.45%
Proposed Approach	**Multiple Classifier + Ensemble Learning**	**eEnvelope + iForest + LOF + OneClassSVM**	**93.53%**	**99.52%**	**100%**

Table 17
Comparing the proposed approach with other researchers' works regarding the accuracy, precision, recall, and f1-score on CCF dataset.

Authors & Ref.	Approach	Base learners	Performance evaluation			
			Accuracy	*Precision*	*Recall*	*F1-Score*
N. S. Alfaiz et al. [18]	Boosting (CatBoost)	KNN + CatBoost	NA	NA	95.91%	87.40%
Y. F. Zhang et al. [21]	Boosting (AdaBoost)	AdaBoost + OCSVM	96.00%	97.00%	96.00%	98.00%
F. Carcillo et al. [22]	Ensemble Learning	iForest + RandomForest	97.20%	98.30%	95.00%	NA
V. Karthik et al. [24]	Bagging + Boosting (AdaBoost)	AdaBoost + RandomForest	99.18%	NA	99.5%	NA
J. Forough et al. [25]	Ensemble Learning	RNN + FFNN	NA	95.69%	66.74%	78.13%
R. Asha et al. [26]	Multiple Classifier	ANN + KNN + SVM	93.49%	97.43%	89.76%	NA
T. A. Olowookere et al. [29]	Ensemble Learning	KNN + DT + MLP	NA	NA	NA	97.40%
S. Khatri et al. [30]	Multiple Classifier	KNN + DT	NA	85.11%	91.11%	NA
S. N. Kalid et al. [31]	Multiple Classifier	C4.5 + NB	99.90%	83.45%	87.20%	NA
V. N. Dornadula et al. [32]	Multiple Classifier	RandomForest	97.08%	98.14%	NA	NA
I. Sohony et al. [33]	Multiple Classifier	RF + FFNN	99.90%	85.85%	86.70%	NA
K. Randhawa et al. [34]	AdaBoost + Majority Voting	ANN + NB	99.90%	NA	78.90%	NA
S. Ram et al. [35]	Ensemble Learning	CNN	96.50%	NA	NA	NA
P. Xenopoulos [36]	Ensemble Learning	Deep Belief Network	90.60%	NA	81.80%	NA
Proposed Approach	**Multiple Classifier + Ensemble Learning**	**eEnvelope + iForest + LOF + OneClassSVM**	**99.98%**	**99.96%**	**100%**	**99.95%**

Table 18
Comparing the proposed approach with other researchers' works regarding the accuracy, precision, recall, and f1-score on CCDP dataset.

Authors & Ref.	Approach	Base learners	Performance evaluation			
			Accuracy	*Precision*	*Recall*	*F1-Score*
E. F. Malik et al. [19]	Boosting (AdaBoost, LGBM)	AdaBoost + LGBM	NA	97.00%	64.00%	77.00%
X. Zhang et al. [23]	Ensemble Learning	BPNN + SVM + RF	NA	96.00%	94.50%	96.02%
M. Seera al. [27]	Multiple Classifier	ANN + SVM	96.49%	NA	NA	NA
F. Itto et al. [28]	Multiple Classifier	NB + KNN + LR	95.00%	97.00%	NA	NA
P. Save et al. [37]	Hunt's and Luhins Algorithms	DT	NA	NA	NA	NA
Xia et al. [38]	Bossting (XGBoost)	DT	69.36%	NA	NA	NA
Pradeep Singh [39]	Bagging	RF	81.60%	NA	37.10%	NA
A. Charleonnan et al. [40]	Multiple Classifier	MLP + RBF + NB	77.80%	NA	NA	NA
Venkatesh et al. [41]	Bagging	RF + NB	NA	NA	81.60%	NA
Proposed Approach	**Multiple Classifier + Ensemble Learning**	**eEnvelope + iForest + LOF + OneClassSVM**	**99.00%**	**100%**	**98.09%**	**99.00%**

Table 19
Complexity analysis.

Algorithms	Time complexity	Statement
Elliptic Envelope [46]	$O(n \cdot m^2)$	n = Number of samples & m = Number of features
Isolation Forest [47]	$O(n \cdot d)$	n = Number of samples & d = Max depth
Local Outlier Factor [49]	$O(n^2 \cdot m)$	n = Number of samples & m = Number of features
One-ClassSVM [52]	$O(n^2 \cdot v)$	n = Number of samples & v = Number of support vectors
XGBoost [54]	$O(n \cdot m \cdot t)$	n = Number of samples, m = Number of features, & t = Number of trees
Algorithm 1	$O(n)$	n = Feature importance score
Algorithm 2	$O(4 \cdot n \cdot t) + O(n) + O(p \cdot q)$	n = Number of samples, t = Train classifier complexity, p = Number of trees & q = Train meta classifier complexity
Algorithm 3	Step 1: $O(s) + O(k \cdot n)$; Step 2: $O(4 \cdot n \cdot t) + O(n) + O(p \cdot q)$; Step 3: $O(r)$	s = Number of data points, k = Number of folds, n = Number of samples t = Train classifier complexity, p = Number of trees, q = Train meta classifier complexity r = Number of test samples
Algorithm 4	$O(n)$	n = Number of elements

better than the existing works, particularly anomaly detection from the minority class of credit card datasets. In our experimental studies, we found that some single classifiers or outlier detection algorithms, i.e., *OneClassSVM*, *eEnvelope*, *iForest*, and *LOF* are suitable for classifying minority and majority class samples. For this reason, in our proposed *CCAD* model, we employ them as *base learners*, and then we utilize *XGBoost* as a *meta-learner* through ensemble to identify anomalies. Besides that, the discordance value calculation assist in constructing the ensemble algorithm more accurate and effective for fraud prediction. This proposed *CCAD* model was assessed utilizing two datasets: (i) *CCF* & (ii) *CCDP*. Moreover, our proposed work has been compared with existing research works, and experimental results show the capability and performance are better than the existing state-of-the-art approaches.

In principle, this *CCAD* model presents its excellence in tackling the issues of *imbalanced class distribution*, *overfitting*, and *overlapping* classes. However, there is still some space to increase the precision value for minority classes, and we are glancing at other ensemble techniques and hybrid architecture for future works. Nowadays, some researchers had tried deep learning methods, i.e., *DBN* and *LSTM*, to identify the anomalies in the transaction data of credit cards. So, to detect promising outcomes, we are also thinking about deep learning methods for our future work. Moreover, we intend to apply our *CCAD* model to other credit card datasets to determine our model's robustness.

Acronym

The full form of the acronym used in paper is presented in Table 20.

Table 20
Explanation of notations.

Notation	Definition
ANN	Artificial Neural Networks
AdaBoost	Adaptive Boosting
AUC	Area Under the Curve
CCAD	Credit Card Anomaly Detection
CCF	Credit Card Fraud
CCDP	Credit Card Default Payment
CART	Classification and Regression Tree
CNN	Convectional Neural Network
DBN	Deep Belief Networks
DL	Deep Learning
eEnvelope	Elliptic Envelope
FPR	False Positive Rate
FP	False Positive
FN	False Negative
FFNN	Feed-Forward Neural Network
GBC	Gradient Boosting
HOBA	Homogeneity-oriented Behavior Analysis
iForest	Isolation Forest
IEEE-CIS	IEEE Computational Intelligence Society
KNN	K-Nearest Neighbors
LR	Logistic Regression
LOF	Local Outlier Factor
LSTM	Long Short-term Memory
MCS	Multiple Classifier Systems
ML	Machine Learning
MLP	Multilayer Perceptron
MCC	Matthews Correlation Coefficient
NB	Naïve Bayes
OCC	One-class classification
OCSVM	One-class Support Vector Machine
CatBoost	Categorical Boosting
PAC	Probably Approximately Correct
PPV	Precision or Positive Predictive Value
RBF	Radial Basis Function
RF	Random Forest
ROC	Receiver Operating Characteristic Curve
RNN	Recurrent Neural Network
SVM	Support Vector Machines
SMOTE	Synthetic Minority Oversampling Technique
TN	True Negative
TNR	True Negative Rate
TPR	True Positive Rate
TP	True Positive
ULB	University Library de Brussels
XGBoost	eXtreme Gradient Boosting

CRediT authorship contribution statement

Md Amirul Islam: Concept of the study and designed it, Performed the simulation and the acquisition of data; Prepare the first draft of the paper. **Md Ashraf Uddin:** Involved in collecting data, Preparing the first draft of the paper, Offering suggestions and comments to improve the study. **Sunil Aryal:** provide the preliminary guidelines to develop, Write the paper. **Giovanni Stea:** reviewed paper, Revised the texts of the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

This work is supported by the Deakin University, Australia, and Air Force Office of Scientific Research under award number FA2386-23-1-4003 and partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the FoReLab project (Departments of Excellence).

References

[1] Dong Q, Gong S, Zhu X. Class rectification hard mining for imbalanced deep learning. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 1851–60.
[2] Gao J, Gong L, Wang J, Mo Z. Study on unbalanced binary classification with unknown misclassification costs. In: 2018 IEEE international conference on industrial engineering and engineering management. IEEE; 2018, p. 1538–42.
[3] MoneyTransferscom. 15 Shocking credit card fraud statistics & facts for 2022. 2022, [Online]. Available: https://moneytransfers.com/news/2022/09/21/credit-card-fraud-statistics. [Accessed 10 April 2023].
[4] Chargebacks911com. Credit card fraud statistics. 2023, [Online]. Available: https://chargebacks911.com/credit-card-fraud-statistics/. [Accessed 10 April 2023].
[5] legaljobsio. 20 Most fascinating credit card fraud statistics. 2022, [Online]. Available: https://legaljobs.io/blog/credit-card-fraud-statistics/. [Accessed 10 April 2023].
[6] Lokman T. 3.6 Million credit card holders have RM36.9 billion outstanding balance. 2017, [Online]. Available: https://www.nst.com.my/news/nation/2017/08/270620/36-million-credit-card-holders-have-rm369-billion-outstanding-balance. [Accessed 10 April 2023].
[7] Ceicdata. Malaysia credit card statistics. 2018, [Online]. Available: https://www.ceicdata.com/en/malaysia/credit-card-statistics. [Accessed 10 April 2023].
[8] Vuttipittayamongkol P, Elyan E, Petrovski A, Jayne C. Overlap-based undersampling for improving imbalanced data classification. In: Intelligent data engineering and automated learning–IDEAL 2018: 19th international conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part I. Vol. 19. Springer; 2018, p. 689–97.
[9] Mathew J, Pang CK, Luo M, Leong WH. Classification of imbalanced data by oversampling in kernel space of support vector machines. IEEE Trans Neural Netw Learn Syst 2017;29(9):4065–76.
[10] Bhattacharyya S, Jha S, Tharakunnel K, Westland JC. Data mining for credit card fraud: A comparative study. Decis Support Syst 2011;50(3):602–13.
[11] Sahin Y, Bulkan S, Duman E. A cost-sensitive decision tree approach for fraud detection. Expert Syst Appl 2013;40(15):5916–23.
[12] Zakaryazad A, Duman E. A profit-driven artificial neural network (ANN) with applications to fraud detection and direct marketing. Neurocomputing 2016;175:121–31.
[13] Jurgovsky J, Granitzer M, Ziegler K, Calabretto S, Portier P-E, He-Guelton L, et al. Sequence classification for credit-card fraud detection. Expert Syst Appl 2018;100:234–45.
[14] Fu K, Cheng D, Tu Y, Zhang L. Credit card fraud detection using convolutional neural networks. In: Neural information processing: 23rd International conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part III. Vol. 23. Springer; 2016, p. 483–90.
[15] Zareapoor M, Shamsolmoali P, et al. Application of credit card fraud detection: Based on bagging ensemble classifier. Procedia Comput Sci 2015;48(2015):679–85.
[16] Rout M. Analysis and comparison of credit card fraud detection using machine learning. In: Advances in electronics, communication and computing: Select proceedings of ETAEERE 2020. Springer; 2021, p. 33–40.
[17] Kim E, Lee J, Shin H, Yang H, Cho S, Nam S-k, et al. Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning. Expert Syst Appl 2019;128:214–24.
[18] Alfaiz NS, Fati SM. Enhanced credit card fraud detection model using machine learning. Electronics 2022;11(4):662.
[19] Malik EF, Khaw KW, Belaton B, Wong WP, Chew X. Credit card fraud detection using a new hybrid machine learning architecture. Mathematics 2022;10(9):1480.
[20] Deepanath C, Prasad K. IEEE-CIS fraud detection. Indian Institute of Management Bangalore; 2019.
[21] Zhang Y-F, Lu H-L, Lin H-F, Qiao X-C, Zheng H. The optimized anomaly detection models based on an approach of dealing with imbalanced dataset for credit card fraud detection. Mob Inf Syst 2022;2022.
[22] Carcillo F, Le Borgne Y-A, Caelen O, Kessaci Y, Oblé F, Bontempi G. Combining unsupervised and supervised learning in credit card fraud detection. Inf Sci 2021;557:317–31.
[23] Zhang X, Han Y, Xu W, Wang Q. HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. Inform Sci 2021;557:302–16.
[24] Karthik V, Mishra A, Reddy US. Credit card fraud detection by modelling behaviour pattern using hybrid ensemble model. Arab J Sci Eng 2021;1–11.
[25] Forough J, Momtazi S. Ensemble of deep sequential models for credit card fraud detection. Appl Soft Comput 2021;99:106883.
[26] Asha R, KR SK. Credit card fraud detection using artificial neural network. Glob Trans Proc 2021;2(1):35–41.
[27] Seera M, Lim CP, Kumar A, Dhamotharan L, Tan KH. An intelligent payment card fraud detection system. Ann Oper Res 2021;1–23.
[28] Itoo F, Singh S. Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection. Int J Inf Technol 2021;13:1503–11.
[29] Olowookere TA, Adewale OS. A framework for detecting credit card fraud with cost-sensitive meta-learning ensemble approach. Sci Afr 2020;8:e00464.
[30] Khatri S, Arora A, Agrawal AP. Supervised machine learning algorithms for credit card fraud detection: A comparison. In: 2020 10th International conference on cloud computing, data science & engineering. IEEE; 2020, p. 680–3.
[31] Kalid SN, Ng K-H, Tong G-K, Khor K-C. A multiple classifiers system for anomaly detection in credit card data with unbalanced and overlapped classes. IEEE Access 2020;8:28210–21.
[32] Dornadula VN, Geetha S. Credit card fraud detection using machine learning algorithms. Procedia Comput Sci 2019;165:631–41.
[33] Sohony I, Pratap R, Nambiar U. Ensemble learning for credit card fraud detection. In: Proceedings of the ACM India joint international conference on data science and management of data. 2018, p. 289–94.
[34] Randhawa K, Loo CK, Seera M, Lim CP, Nandi AK. Credit card fraud detection using AdaBoost and majority voting. IEEE Access 2018;6:14277–84.
[35] Ram S, Gupta S, Agarwal B. Devanagri character recognition model using deep convolution neural network. J Stat Manag Syst 2018;21(4):593–9.
[36] Xenopoulos P. Introducing DeepBalance: Random deep belief network ensembles to address class imbalance. In: 2017 IEEE international conference on big data. IEEE; 2017, p. 3684–9.
[37] Save P, Tiwarekar P, Jain KN, Mahyavanshi N. A novel idea for credit card fraud detection using decision tree. Int J Comput Appl 2017;161(13).
[38] Xia Y, Liu C, Li Y, Liu N. A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. Expert Syst Appl 2017;78:225–41.
[39] Singh P. Comparative study of individual and ensemble methods of classification for credit scoring. In: 2017 International conference on inventive computing and informatics. IEEE; 2017, p. 968–72.
[40] Charleonnan A. Credit card fraud detection using RUS and MRN algorithms. In: 2016 Management and innovation technology international conference. IEEE; 2016, p. MIT–73.
[41] Venkatesh A, Jacob SG. Prediction of credit-card defaulters: A comparative study on performance of classifiers. Int J Comput Appl 2016;145(7).
[42] Carneiro N, Figueira G, Costa M. A data mining based system for credit-card fraud detection in e-tail. Decis Support Syst 2017;95:91–101.
[43] Yeh I, Lien C. Default of credit card clients data set. 2016, Verkregen van archive. ics. uci. edu/ml/datasets/default+ of+ credit+ card+ clients.
[44] Li T, Chen X-r, Tang H, Xu X-k. Identification of the normal/abnormal heart sounds based on energy features and Xgboost. In: Chinese conference on biometric recognition. Springer; 2018, p. 536–44.
[45] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.
[46] Hoyle B, Rau MM, Paech K, Bonnett C, Seitz S, Weller J. Anomaly detection for machine learning redshifts applied to SDSS galaxies. Mon Not R Astron Soc 2015;452(4):4183–94.
[47] Liu FT, Ting KM, Zhou Z-H. Isolation forest. In: 2008 Eighth IEEE international conference on data mining. IEEE; 2008, p. 413–22.
[48] Cheng Z, Zou C, Dong J. Outlier detection using isolation forest and local outlier factor. In: Proceedings of the conference on research in adaptive and convergent systems. 2019, p. 161–8.
[49] Breunig MM, Kriegel H-P, Ng RT, Sander J. LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data. 2000, p. 93–104.
[50] Wang L. Support vector machines: Theory and applications. Vol. 177. Springer Science & Business Media; 2005.
[51] Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM Comput Surv (CSUR) 2009;41(3):1–58.
[52] Schölkopf B, Williamson RC, Smola A, Shawe-Taylor J, Platt J. Support vector method for novelty detection. In: Advances in neural information processing systems. Vol. 12. 1999.
[53] Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC. Estimating the support of a high-dimensional distribution. Neural Comput 2001;13(7):1443–71.
[54] Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining. 2016, p. 785–94.
[55] Trisanto D, Rismawati N, Mulya MF, Kurniadi FI. Modified focal loss in imbalanced XGBoost for credit card fraud detection. Int J Intell Eng Syst 2021;14:350–8.
[56] Wang C, Deng C, Wang S. Imbalance-XGBoost: Leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. Pattern Recognit Lett 2020;136:190–7.
[57] Da Silva NF, Hruschka ER, Hruschka Jr. ER. Tweet sentiment analysis with classifier ensembles. Decis Support Syst 2014;66:170–9.

[58] Bagga S, Goyal A, Gupta N, Goyal A. Credit card fraud detection using pipeling and ensemble learning. Procedia Comput Sci 2020;173:104–12.
[59] Feng H. Ensemble learning in credit card fraud detection using boosting methods. In: 2021 2nd International conference on computing and data science. IEEE; 2021, p. 7–11.
[60] Flennerhag S, Moreno PG, Lawrence ND, Damianou A. Transferring knowledge across learning processes. 2018, arXiv preprint arXiv:1812.01054.
[61] Bey R, Goussault R, Grolleau F, Benchoufi M, Porcher R. Fold-stratified cross-validation for unbiased and privacy-preserving federated learning. J Am Med Inf Assoc 2020;27(8):1244–51.
[62] Dal Pozzolo A, Caelen O, Johnson RA, Bontempi G. Calibrating probability with undersampling for unbalanced classification. In: 2015 IEEE symposium series on computational intelligence. IEEE; 2015, p. 159–66.
[63] Vishwakarma SK, Rasool A, Hajela G. Machine learning algorithms for prediction of credit card defaulters—A comparative study. In: Proceedings of international conference on sustainable expert systems. Springer; 2021, p. 141–9.
[64] Powers DM. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2020, arXiv preprint arXiv:2010.16061.
[65] Zhenya Q, Zhang Z. A hybrid cost-sensitive ensemble for heart disease prediction. BMC Med Inf Decis Mak 2021;21:1–18.
[66] He H, Garcia EA. Learning from imbalanced data. IEEE Trans Knowl Data Eng 2009;21(9):1263–84.
[67] Raghuwanshi BS, Shukla S. Underbagging based reduced kernelized weighted extreme learning machine for class imbalance learning. Eng Appl Artif Intell 2018;74:252–70.
[68] Chicco D, Tötsch N, Jurman G. The matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. BioData Min 2021;14(1):1–22.
[69] Khaldy M, Kambhampati C. Resampling imbalanced class and the effectiveness of feature selection methods for heart failure dataset. Int Robot Autom J 2018;4(1):1–10.
[70] Müller AC, Guido S. Introduction to machine learning with python: A guide for data scientists. O'Reilly Media, Inc; 2016.
[71] Fawcett T. An introduction to ROC analysis. Pattern Recognit Lett 2006;27(8):861–74.
[72] Carta S, Fenu G, Recupero DR, Saia R. Fraud detection for E-commerce transactions by employing a prudential Multiple Consensus model. J Inf Secur Appl 2019;46:13–22.