An Ensemble-Based Approach for Cyber Attack Detection in Financial Systems

[1]Dhanisht Kumar, [1]Harshit Srivastava, [1]Kartik S. Bhamare, [2]Malay Kumar
[1]B.Tech. Students, [2]Assistant Professor
[1,2]Department of Computer Science and Engineering
[1,2]Indian Institute of Information Technology Dharwad

Abstract- Digital commerce is the process of exchanging money or assets using electronic or digital channels such as the internet or mobile devices. Instead of physical cash or checks, individuals or businesses can use digital platforms to transfer money, make payments or conduct financial transactions securely and efficiently. As more and more people conduct financial transactions online, the risk of cyberattacks targeting financial institutions will increase. This article presents an integrated approach to detecting cyber-attacks in the financial sector. The proposed method combines learning models including decision trees, random forests and gradient boosting to increase the accuracy and robustness of the model.

Keywords: Cyber Attack, Digital commerce, Financial Systems, Fraud, Machine Learning

1. Introduction

In the past, cash was often used to exchange goods and services due to the lack of network resources and smart device support [1]. However, with the advancement in software, networks and hardware (such as 5G networks) and the increasing use of mobile phones, we now have the necessary tools to help the businesses digitize. As a result, business technology has evolved into a digital model [2]. While the transition to digital commerce has brought benefits such as convenience and speed, it has also created new opportunities for cyber attackers to exploit security vulnerabilities in financial systems. Therefore, there is an urgent need to develop effective methods to detect and prevent cyber-attacks on financial markets. As the technology advances, more and more payment methods are added to financial transactions, this article proposes a novel algorithm for detecting cyber-attacks in all types of transactions such as ***Unified Payment Interface (UPI)***, ***Online Banking***, ***Mobile Wallets***, ***Credit and Debit Cards***, ***Aadhar-Enabled Payment System*** (AEPS), ***USSD*** etc. The experiment has been performed on credit card data only due to unavailability of fraudulent transaction data on other payment systems. In recent years, machine learning techniques have shown promise results in detecting and preventing cyberattacks. In particular, hybrid methods combining multiple models to improve performance are appreciated for their ability to solve problems such as overfitting and bias [3].

2. Background

2.1. Digital Transaction

The origins of electronic payments can be traced back to **'1871'**, when "Western Union" pioneered the concept of Electronic Funds Transfer (EFT) in the United States [5]. The boom marked a turning point in the history of the economy, allowing people to purchase goods and services without being physically present in the market. The launch of EFT paved the way for a new era of electronic commerce, replacing traditional payment methods and ushering in a new era of convenience and efficiency. The concept of EFT is a technological change that will make money transfers longer and more secure. After the development of ARPANET (1960), digital payments gained another momentum. In 1994, Stanford Federal Credit Union introduced its online banking service [3] and other banks did the same, finally allowing their customers to log in to online payments in 1995. In the year **1996** the Credit and Investment Corporation of India (ICICI Bank Ltd.) become the first bank in India to offer online banking services to its customers [3]. However, it wasn't the year **2008** that the National Payments Corporation of India (NPCI) established a better payment system, paving the way for many new products and services in India for digital payment systems. More importantly, this period also saw the 2010 launch of Aadhaar, India's unique digital number system, which has become an integral part of the country's digital ecosystem [4].

2.2. Digital Payment Methods

A. ***Banking Cards (Credit/Debit):*** Credit/Debit Cards are plastic cards with magnetic strip or chip on card, which holds the details of the card owner. Credit/Debit cards are modern ways of financial transactions [4].
B. ***Unstructured Supplementary Service Data (USSD):*** USSD is an India specific payment service, which work on Unstructured Supplementary Service Data (USSD) channel. This service allows financial transaction using basic feature phones.

C. ***Aadhar Enabled Payment System (AEPS):*** AEPS is a digital payment model which allows online interoperable financial transactions connected with India's AADHAAR Card Network [4].

D. ***Unified Payment Interface (UPI):*** UPI provides a common payment interface for most of the banks doing business in India. UPI facilitates to perform digital transactions upto the limit of Rs. 1 lacs daily over mobile applications such as Paytm, Phonepay, Google Pay, BHIM and others.

E. ***Mobile Wallets:*** A mobile wallet is a digital wallet designed to store money in electronic form, providing a convenient and secure means of carrying cash digitally. It functions as an online payment solution by establishing a connection between its application and debit or credit cards [4].

F. ***Internet Banking:*** It enables e-banking or virtual banking service to the customer's belonging banks or financial institutions [4].

G. ***Mobile Banking:*** Mobile banking is a convenient service offered by banks and financial institutions, enabling customers to perform various financial transactions remotely through their mobile devices, such as smartphones. Prominent examples of mobile banking applications include YONO SBI, ICICI iMobile App, IDBI, and several others [4].

2.3. Framework for Online Transaction System

The general framework for online transaction systems is presented in Figure 1. The process begins with a user accessing a payment gateway to initiate a transaction. The payment processor transmits the transaction details to the payment processing network (PPN) for authorization. Upon receiving authorization, the issuing bank assesses the availability of funds and approves or declines the transaction accordingly. Subsequently, the transaction flows through the electronic or mobile network to the processor, and an approval code is transmitted to the sales device at the merchant's location. The issuing bank then transfers the funds to the processing company as reimbursement for the completed purchase. Finally, the processor deposits the merchant's funds into their bank account.

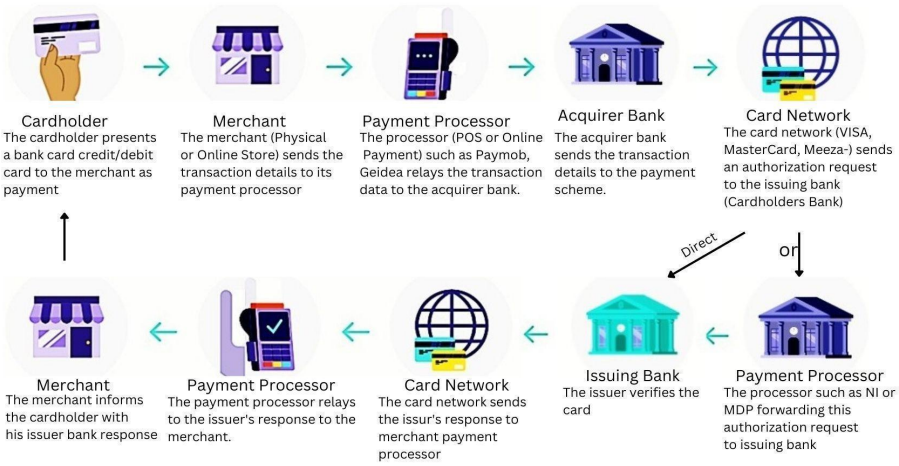

Fig. 1: General Framework for Transaction Systems

2.4. Digital Payment Fraud

According to the Federal Trade Commission, the total economic loss in the United States for 2022 is $8.8 billion [5]. India lacks reporting from any authentic financial institution. However, the media reports many cases of financial fraud. According to the Economic Times report, a total of 1.4 corer fraud cases were reported in the 2022 fiscal year, including credit card and UPI fraud. However, in our research, we did not find any publicly available report or data on credit card or UPI fraud specific from Indian financial institutions.

2.4.1. Types of Fraud

A. ***Identity theft:*** Identity theft occurs when an individual utilizes another person's personal information, such as their Social Security number or credit card details, with the intention of engaging in fraudulent activities. In the year 2021 alone, there were over 1.4 million reported cases of identity theft.

B. ***Skimming:*** Skimming occurs when criminals illicitly install devices on ATMs to unlawfully capture card numbers and PINs. The act of skimming results in significant financial losses, estimated to exceed $1 billion annually.

C. ***Phishing:*** Phishing is a deceptive technique employed by fraudsters to deceive unsuspecting individuals into divulging sensitive information, including usernames, passwords, and credit card numbers. This fraudulent activity is typically carried out through the use of emails or text messages.

D. ***Account Takeover:*** Account takeover transpires when a criminal gains control of another individual's online account using pilfered credentials, such as a stolen username or password. Typically, these illicit credentials are obtained through purchases made on the Dark Web.

E. ***Man-in-the-Middle Attacks:*** In this type of attack, the attacker intercepts the communication between two parties and steals sensitive information such as account numbers and passwords.

F. ***Denial of Service (DoS) Attacks:*** These attacks are designed to overload a financial institution's servers, making it impossible for customers to carry out financial transactions.

G. ***Log-Based Attack:*** A log-based attack is a sophisticated technique utilized by cybercriminals to infiltrate the logging infrastructure of financial institutions or payment systems to steal sensitive information such as credit card details, login credentials, and transaction records.

3. Related work

Currently, many machine learning (ML) algorithms are used to identify vulnerabilities in financial systems. Fig. 2 illustrates the latest developments in this field, showing machine learning-based methods for identifying vulnerabilities. The tree structure (Figure 2) shows the various algorithms used and their relative accuracy levels and provides an overview of the current state of vision security. The classification of the research attack is shown in fig.2. Interested readers can refer to the article for more details.

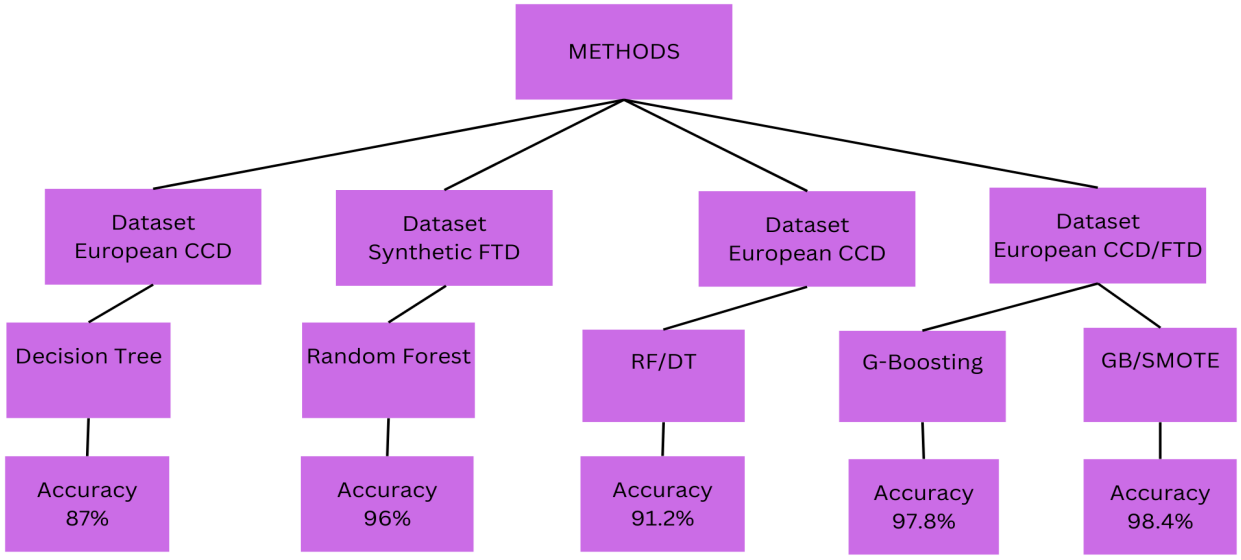

Fig. 2: Taxonomy of Attack Detection Methods

4 Methodology

4.1 Operational procedure flow

The work includes data collection, preprocessing and modeling, which includes collecting and pre-processing relevant data and transforming the data into research-appropriate data. The entire process flow is shown in Fig. 3 below.

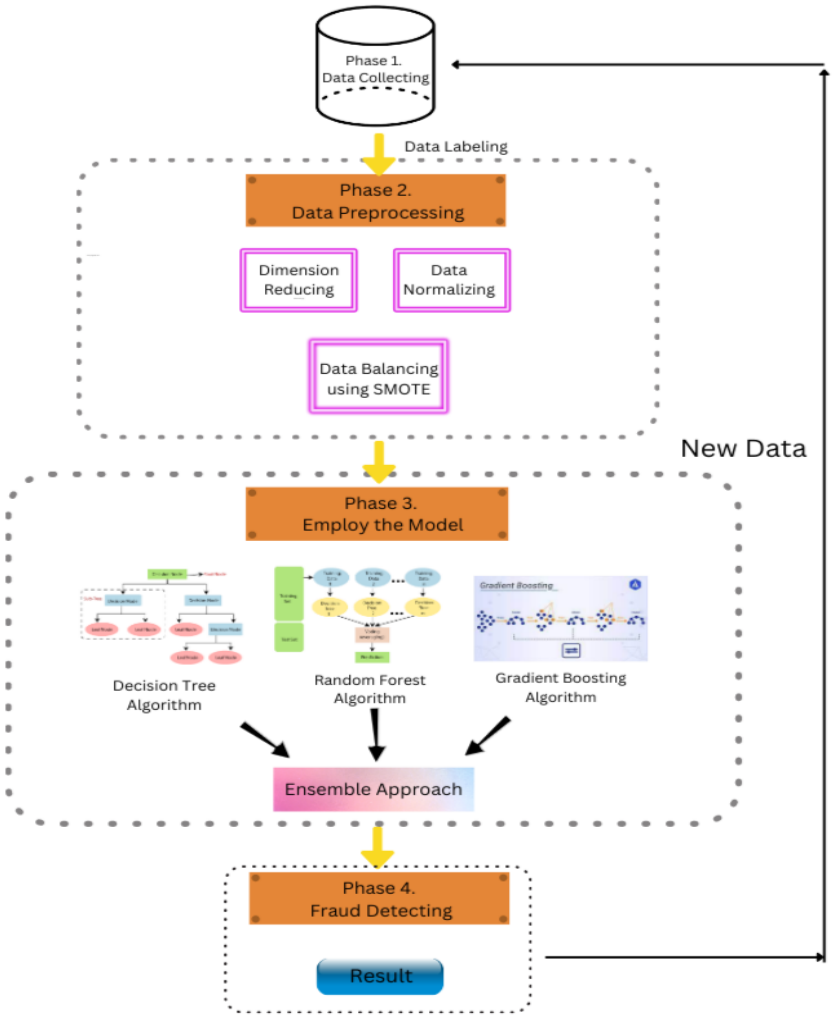

Fig. 3: Process flow diagram

- **Dataset:** PaySim synthetic dataset has been included for study, features including transaction type, amount, customer names, balances, cash-in, cash-out, and attack indicator class.
- **Preprocessing:** Data preprocessing steps involves, data cleaning, data normalization and importantly data balancing. To mitigate class imbalance, we applied the SMOTE algorithm, which generates synthetic minority-class instances to improve representation and balance the number of instances in each class. Take a closer look at the SMOTE algorithm below with careful consideration. After applying SMOTE the plotting of imbalance and balanced data has been shown in Fig 4(a) and Fig 4(b) respectively.

SMOTE: Before diving into the SMOTE we should have understanding of why we select SMOTE oversampling technique to balance the dataset. It's because, this technique creates synthetic samples for the minority class by generating new instances between existing samples, helping the model learn the minority class better and improving overall prediction accuracy, also it does not lose any single data.

SMOTE: Algorithm

Oversampling:

Input: *Imbalanced data*
- Minority class examples X_min
- Number of synthetic examples to generate N
- Number of nearest neighbors k

Intermediate result:
- New synthetic examples X_syn

Initialize *X_syn as an empty set*

For *each x in X_min:*
| *Find the k nearest neighbors of x from X_min (excluding x itself)*
| ***For*** *i from 1 to N:*
| | *Choose one of the k nearest neighbors randomly and call it x'*
| | *Generate a new synthetic example as follows:*
| | $X\_new = x + rand(0, 1) * (x' - x)$
| | ***Add*** *X_new to X_syn*

Return *X_syn*

Output: *Balanced data*

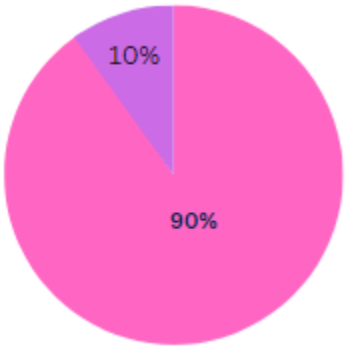

Fig: 1-Imbalanced

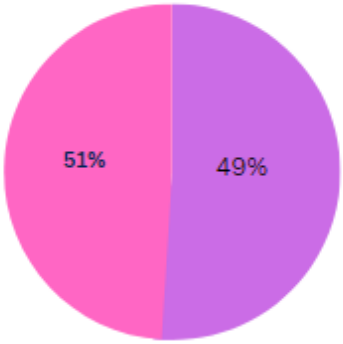

Fig: 2 - Balanced

Fig: 1 represents the dataset is quite unbalanced before applying SMOTE oversampling technique.

Fig: 2 represents the dataset is balanced by applying SMOTE oversampling technique.

Distribution of class labels in dataset:

$$H(S) = -\Sigma\, p(i)\, log2\, p(i) \quad (1)$$

$$IG(S, A) = H(S) - \Sigma\, (|S| / |S|)\, H(S) \quad (2)$$

Equation (1): It calculates the entropy of a dataset S. It measures the uncertainty or disorder in the distribution of class labels within S. The higher the entropy, the more mixed the class labels are, and vice versa.

Information Gain Formula (IG(S, A)): Equation (2) computes the information gain of attribute A in relation to dataset S. It quantifies how much splitting the data based on attribute A reduces the overall entropy. A

higher information gain implies that attribute A is more useful for distinguishing classes within the dataset. Carrying this information about data to the training algorithms.

Table 1: Notation & Parameters

Notations	Identification
$H(S)$	Entropy of Extracted Data
$P(i)$	Probability of occurrence of class i
$IG(S, A)$	Information gain after Ensemble

Algorithm 1: Decision Tree

Input: *Training dataset D*
Features F
Maximum depth max_depth

Output: *Decision tree model*

function *build_decision_tree(D, F, depth):*
| ***if*** *depth > max_depth or all samples in D belong to the same class:*
| | *return LeafNode(majority_class(D))*
| ***else:***
| | ***Select*** *the best feature F_best to split D based on some criterion*
| | *Split D into subsets D1, D2, ..., Dk based on the values of F_best*
| | *node = InternalNode(feature=F_best)*
| | ***for*** *each subset Di:*
| | | ***if*** *Di is not empty:*
| | | | *child_node = build_decision_tree(Di, F, depth + 1)*
| | | | *node.add_child(child_node, value=Fi)*
| | ***return*** *node*

root_node *=* ***build_decision_tree****(D, F, depth=1)*
return *DecisionTree(model=root_node)*

Algorithm 2 Random Forest model

Input: *Training dataset D, Number of trees num_trees, Maximum depth max_depth, Number of features k_features*
Output: *Random Forest model*
function *calculate_gini(D):*
| *Calculate class distribution frequencies*
| $Gini = 1 - \Sigma (p\_i)\text{^}2 \; for \; each \; class \; i$

```
|  return Gini
function calculate_split_gini(D, F):
|  Split D into subsets D1 and D2 by feature F
|  Split_gini = (|D1| / |D|) * calculate_gini(D1) + (|D2| / |D|) * calculate_gini(D2)
|  return Split_gini
function select_best_feature(D, features):
|  best_feature, best_split_gini = None, ∞
|  for each feature F in features:
|  |  split_gini = calculate_split_gini(D, F)
|  |  if split_gini < best_split_gini:
|  |  |  best_split_gini, best_feature = split_gini, F
|  return best_feature
function build_tree(D, features, depth):
|  if depth > max_depth or samples in D are same class:
|     return LeafNode(majority_class(D))
|  else:
|  |  best_feature = select_best_feature(D, features)
|  |  node = InternalNode(feature=best_feature)
|  |  for each subset Di:
|  |  |  if Di is not empty:
|  |  |  |  child_node = build_tree(Di, features, depth + 1)
|  |  |  |  node.add_child(child_node, value=Fi)
|  return node
function build_forest(D, num_trees, max_depth, k_features):
|  forest = []
|  for i = 1 to num_trees:
|  |  D_subset = Randomly sample subset of D
|  |  tree = build_tree(D_subset, k_features, max_depth)
|  |  forest.append(tree)
|  return RandomForest(model=forest)
training_data = LoadTrainingData()
num_trees = 10, max_depth = 5, k_features = √(number_of_features)
random_forest_model = build_forest(training_data, num_trees, max_depth, k_features)
return random_forest_model
```

Algorithm 3 Gradient Boosting

```
Input: Training dataset D, Number of boosting iterations num_iterations, Learning rate eta
Output: Gradient Boosting model
function initialize_predictions(D):
|  Initialize predictions as zeros: F_0(x) = 0
|  return predictions
function calculate_residuals(y_true, predictions):
|  Calculate residuals: r_i = y_i − F_{i − 1}(x_i)
|  return residuals
function build_weak_learner(D, residuals):
```

| *Fit a weak learner (e.g., decision tree) to minimize residuals*
| ***return*** *fitted_weak_learner*
function *update_predictions(predictions, fitted_weak_learner, eta):*
| $Update\ predictions: F\_i(x) = F\_\{i-1\}(x) + eta * h\_i(x)$
| ***return*** *updated_predictions*
function *build_gradient_boosting(D, num_iterations, eta):*
| *predictions = initialize_predictions(D)*
| ***for*** *i = 1 to num_iterations:*
| | *residuals = calculate_residuals(y_true, predictions)*
| | *fitted_weak_learner = build_weak_learner(D, residuals)*
| | *updated_predictions = update_predictions(predictions, fitted_weak_learner, eta)*
| ***return*** *GradientBoosting(model=updated_predictions)*
training_data *= LoadTrainingData()*
num_iterations = 100, learning_rate = 0.1
gradient_boosting_model = build_gradient_boosting(training_data, num_iterations, learning_rate)
return *gradient_boosting_model*

The proposed ensemble method integrates decision trees, random forests, and gradient boosting, from different implemented phases. Decision trees establish a hierarchical structure for making decisions based on input features, while random forests aggregate multiple trees to improve accuracy. Gradient boosting iteratively adjusts the model by minimizing residual errors from previous iterations. This combined approach enables the ensemble model to effectively detect fraudulent activity in financial transactions. One of the pivotal components of this ensemble method is gradient boosting, a technique that fine-tunes the model iteratively by minimizing residual errors from previous iterations. In the context of fraud detection, gradient boosting starts with a base model and then sequentially adds subsequent models, each focusing on correcting the errors made by the previous ones. This is achieved by training each new model on the residuals (the differences between predicted and actual values) of the previous model. By gradually improving predictions with each iteration, gradient boosting effectively enhances the model accuracy.

ENSEMBLE Algorithm

Input:
- Training data X_train and y_train
- Number of decision trees T
- Maximum tree depth d
- Number of features to consider at each split f
- Learning rate for gradient boosting eta
- Random seed for reproducibility

Output of Decision Tree:
- Ensemble model consisting of T decision trees

Initialize *an empty list of decision trees*

Pick *from 1 to T:*
| *Randomly select a subset of features of size f*
| *Fit a decision tree of maximum depth d on X_train and y_train, using the selected subset of features*
| *Add the decision tree to the list of trees*

Define *a function predict(x) that takes an input feature vector x and returns the ensemble prediction:*
| ***Initialize*** *an empty list of predictions preds*
| ***For*** *each decision tree in the list of trees:*
| | ***Compute*** *the predicted value for x using the decision tree*
| | *Add the predicted value to the list of predictions preds*
| ***Compute*** *the ensemble prediction by taking the weighted average of the predictions:*
| *y_pred = sum(preds) / T*

Define a function *train(X, y) that takes a training set X and corresponding labels y, and trains the ensemble model:*
| *Fit a gradient boosting model on X_train and y_train, with learning rate eta and the ensemble prediction function as the base learner*
| ***Return*** *the trained ensemble model*

5. Result and Analysis

Evaluating a model's performance based solely on accuracy may leave some information about model effectiveness. Metrics such as ROC score, TP/FP rates, and F1-score should also be considered for better interpretability. Assessing multiple performance factors enables a comprehensive evaluation of the model's ability to accurately classify target outcomes. The ROC score serves as an evaluation metric for determining a model's effectiveness in distinguishing between positive and negative classes. A higher ROC score indicates superior performance of the model in accurately classifying positive and negative instances. In Table. 2, the ROC scores for both the separate and ensemble approaches are summarized, providing an overview of their respective performance in terms of classification accuracy.

5.1 Formulation:

$$OOB\ error\ =\ 1\ /\ n\ \Sigma\ (y\ -\ f(x))\text{^}2, \quad (3)$$

Out-of-Bag Error Formula (OOB error): Equation (3) defines the out-of-bag error for a predictive model. It calculates the average squared difference between the actual target values (y) and the model's predictions (f(x)) for the instances not included in the bootstrap sample. This error metric helps assess how well the model generalizes to new, unseen data.

Table 1: Notation & Parameters

Notations	Identification
$f(x)$	Predicted label by corresponding Tree
y	Actual label
n	Sample size

Precision: Precision of the machine learning model describes how sensible our model is against false positive case.

$$Precision\ =\ TP/(TP\ +\ FP)$$

Recall: Recall of the machine learning model describes how sensible our model is against false negative case. As there is always tradeoff between precision and recall, In this system we want to maximize the recall value as our model aims to avoid any instances where it incorrectly predicts a positive case as negative (false negatives). The focus is on ensuring that all actual positive cases are accurately identified and that no positive cases are overlooked. So consequently we have to minimize the FN value.

$$Recall\ =\ TP/(TP\ +\ FN)$$

$$F1\,Score\ =\ 2\ *\ (Precision\ *\ Recall)\ /\ (Precision\ +\ Recall)$$

$$Accuracy\ =\ (TP\ +\ TN)/(TP\ +\ TN\ +\ FP\ +\ FN)$$

Table 2: Score Matrix

Method	Precision	F1 Score	Sensitivity	Accuracy
Decision Tree	97.22	97.90	98.59	98.16
Random Forest	96.77	97.56	98.36	97.90
Gradient Boosting	98.04	98.52	99.01	98.79
Ensemble	99.24	99.24	99.62	99.65

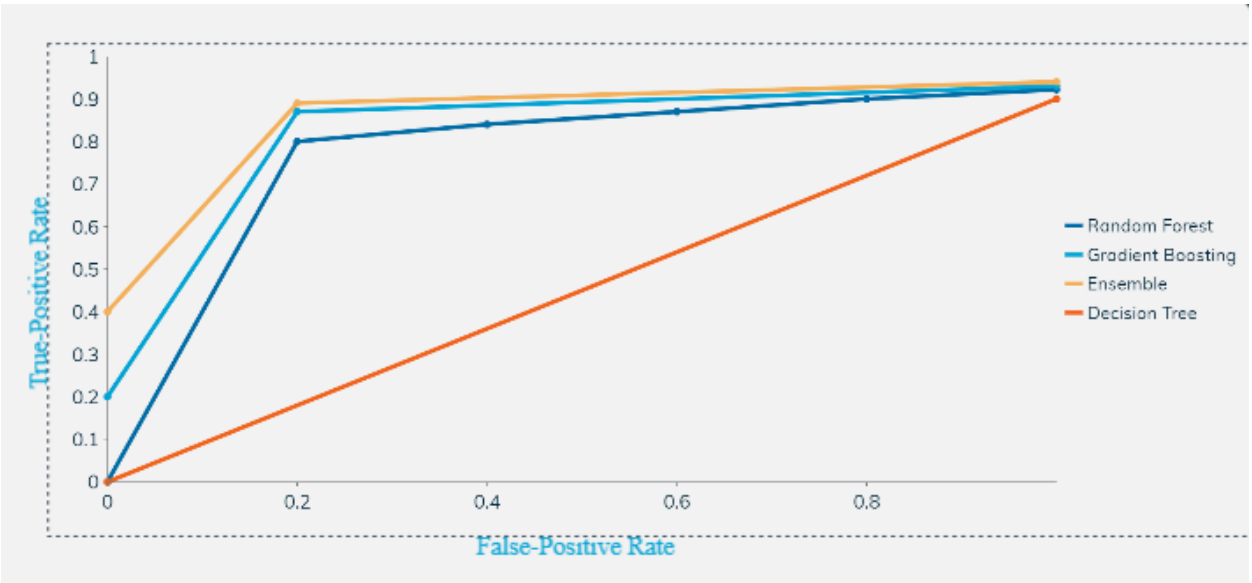

Fig 5: True-Positive ROC Curve

$$Weight\ of\ Model\ I\ =\ (1 - error\ rate\ of\ model\ i)/(Sum\ of\ weight\ of\ all\ models)$$

Here, the error rate of model i is calculated on the validation set, and the weights of all models are summed up to normalize the weights to add up to one. The higher the performance of a model, the higher its weight in the ensemble.

6 Conclusion

The paper proposes an ensemble-based approach using machine learning models to detect cyber-attacks in financial transactions, improving accuracy and robustness by combining diverse models. The research offers a comprehensive overview of digital transactions and the need for effective detection and prevention of cyber-attacks, providing a general framework for online transaction systems. The proposed approach offers a reliable and effective solution to

detect cyber-attacks on financial systems, emphasizing the importance of developing effective methods in the current digital era.

References

[1] https://en.wikipedia.org/wiki/Stanford_Federal_Credit_Union (accessed on 13 Jan, 2023)

[2] Digital Payment Methods." Ministry of Electronics and Information Technology, Government of India. http://cashlessindia.gov.in/digital_payment_methods.html (accessed on 13 Jan, 2023).

[3] Federal Trade Commission. "New FTC Data Show Consumers Reported Losing Nearly $88 Billion to Scams in 2022." Press release, 28 Feb. 2023, https://www.ftc.gov/news-events/news/press-releases/2023/02/new-ftc-data-show-consumers-reported-losing-nearly-88-billion-scams-2022.

[4] Alarfaj, F.K., Malik, I., Khan, H.U., Almusallam, N., Ramzan, M., & Ahmed, M. (2021). Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms. IEEE Access, 9, 50090-50100. DOI: 10.1109/ACCESS.2021.3061626. (GB)

[5] Zhang, X., Han, Y., Xu, W., & Wang, Q. (2019). HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. Knowledge-Based Systems, 161, 145-154. doi: 10.1016/j.knosys.2018.09.017

[6] Taha, A.A. and Malebary, S.J. (2021). An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine. IEEE Access, 9, 51989-51999. DOI: 10.1109/ACCESS.2021.3072879.

[7] Han, S., Zhu, K., Zhou, M., and Cai, X. (2020). Competition-Driven Multimodal Multiobjective Optimization and Its Application to Feature Selection for Credit Card Fraud Detection. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50(9), 3369-3381. DOI: 10.1109/TSMC.2019.2926782.

[8] Zhang, X., Han, Y., Xu, W., and Wang, Q. (2020). HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. Expert Systems with Applications, 142, 113068. DOI: 10.1016/j.eswa.2019.113068.

[9] Carcillo, F., Le Borgne, Y. A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2018). Combining unsupervised and supervised learning in credit card fraud detection. Neurocomputing, 275, 237-245. doi: 10.1016/j.neucom.2017.08.063

[10] Cherif, A., Badhib, A., Ammar, H., Alshehri, S., Kalkatawi, M., & Imine, A. (2021). Credit card fraud detection in the era of disruptive technologies: A systematic review. Journal of Retailing and Consumer Services, 61, 102570. doi: 10.1016/j.jretconser.2021.102570

[11] Forough, J., & Momtazi, S. (2019). Ensemble of deep sequential models for credit card fraud detection. Expert Systems with Applications, 132, 67-80. doi: 10.1016/j.eswa.2019.04.038

[12] Fathy, M., Elhoseny, M., Alkinani, H. H., & Hassanien, A. E. (2021). A Proposed Model for Fraud Detection Based on CatBoost and Deep Neural Network. Journal of King Saud University - Computer and Information Sciences, 33(2), 185-195.

[13] B., Li, L., Li, S., Li, Y., & Li, W. (2021). An Intelligent Approach to Financial Fraud Detection Using an Optimized Light Gradient Boosting Machine. Journal of Computational Science, 51, 101331

[14] Shen, J., Chen, Y., He, X., & Wang, L. (2019). Deep Representation Learning With Full Center Loss for Fraud Detection. IEEE Access, 7, 177342-177352.

[15] Osegi, E. N., & Jumbo, E. F. (2021). Comparative analysis of credit card fraud detection in simulated annealing trained artificial neural network and hierarchical temporal memory. Heliyon, 7(4), e06822. doi: 10.1016/j.heliyon.2021.e06822

[16] Zou, H. (2021). Analysis of best sampling strategy in credit card fraud detection using machine learning. IEEE Access, 9, 22470-22479. doi: 10.1109/ACCESS.2021.3059752

[17] Nguyen, N., Duong, T., Chau, T., Nguyen, V. H., Trinh, T., Tran, D., & Ho, T. (2020). A proposed model for card fraud detection based on CatBoost and deep neural network. Journal of Computational Science, 43, 101171. doi: 10.1016/j.jocs.2020.101171

[18] Zheng, L., Liu, G., Yan, C., Jiang, C., Zhou, M., & Li, M. (2019). Improved TrAdaBoost and Its Application to Transaction Fraud Detection. IEEE Transactions on Fuzzy Systems, 27(12), 2362-2373. doi: 10.1109/TFUZZ.2019.2903873

[19] Fanai, H., & Abbasimehr, H. (2021). A novel combined approach based on deep autoencoder and deep classifiers for credit card fraud detection. Expert Systems with Applications, 166, 114123. doi: 10.1016/j.eswa.2020.114123.

[20] Can, B., Yavuz, A. G., Karsligil, E. M., & Guvensan, M. A. (2020). A closer look into the characteristics of fraudulent card transactions. IEEE Transactions on Information Forensics and Security, 11(11), 2441-2455. doi: 10.1109/TIFS.2016.2598826.

[21] Kalid, S. N., Ng, K. H., Tong, G. K., & Khor, K. C. (2020). A multiple classifiers system for anomaly detection in credit card data with unbalanced and overlapped classes. Expert Systems with Applications, 141, 112998. doi: 10.1016/j.eswa.2019.112998.

[22] Esenogho, E., Mienye, I. D., Swart, T. G., Aruleba, K., & Obaido, G. (2022). A neural network ensemble with feature engineering for improved credit card fraud detection. IEEE Access, 6, 16418-16428. doi: 10.1109/ACCESS.2018.2812039

[23] Nguyen, N., Duong, T., Chau, T., Nguyen, V., Trinh, T., Tran, D., & Ho, T. (2021). A Proposed Model for Card Fraud Detection Based on CatBoost and Deep Neural Network. Sensors, 21(7), 2531. doi: 10.3390/s21072531

[24] Zhu, F., Zhang, C., Zheng, Z., & Al Otaibi, S. (2022). Click Fraud Detection of Online Advertising-LSH Based Tensor Recovery Mechanism. IEEE Access, 8, 100675-100686. doi: 10.1109/access.2020.2993871

[25] Han, S., Zhu, K., Zhou, M., & Cai, X. (2022). Competition-Driven Multimodal Multiobjective Optimization and Its Application to Feature Selection for Credit Card Fraud Detection. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52(12), 7845-7856. doi: 10.1109/TSMC.2021.3111468.