Using Machine Learning Algorithms for Advanced Credit Card Fraud Detection

Fadya Abdulfattah Habeeb
Department of Mathematics, College of Education for Women, Tikrit University, 34001 Tikrit, Salah al-Din, Iraq
Fadya.habeeb@tu.edu.iq

Keywords: Credit Card (CC), Fraud Detection, Predictive Modeling, Machine Learning (ML).

Abstract: Credit card fraud (CCF) is a serious problem that impacts a number of industries, including banking, e-commerce, insurance, finance, commercial entities, and private citizens. More sophisticated technologies are needed for effective detection as fraudulent activities become more complex. Machine learning (ML) approaches have demonstrated high performance in CCF identification. In this study, we explore six ML techniques: CatBoost, Hist Gradient Boosting, Decision Tree (DT), LightGBM (LGBM), ANN, and XGBoost (XGBC). To represent the data, identify its key characteristics, and compare their performance to determine the most successful fraud detection algorithm. Among these algorithms, the CatBoost algorithm, known for its high accuracy and superior performance, stands out. The proposed approach was based on the Kaggle dataset, and the results showed that CatBoost achieved the highest performance with a lower error rate compared to other models. This study also discusses the challenges related to implementing these advanced techniques, including the requirement for sizable, high-quality datasets and significant computational resources for model training and deployment. Key performance measures were used to evaluate the techniques' effectiveness, and CatBoost achieved a test accuracy of 0.9966, outperforming Hist Gradient Boosting, DT, LGBM, XGBC and ANN.

1 INTRODUCTION

The challenge when someone conducts an illicit transaction using a credit card (CC) number, it is referred to as credit card fraud (CCF). Fraud may be caused by a credit card that has been lost, stolen, or is fraudulent. Because more people are making purchases online, fraud involving a card that is not present during online transactions has also increased. The rise of e-banking and other online payment systems has led to a spike in fraud, especially CCF, which causes billions of dollars in losses per year. Identifying CCF has become a critical objective in the era of digital payments [1]. In 2021, credit cards accounted up the largest portion of payments (28%). This implies that CC payments are growing in popularity, especially among higher-income individuals. Additionally, 36% of Americans prefer to use actual or virtual credit cards, whereas 9% of Americans typically utilize cash [2]. Debit and CC payments are increasingly highly valued For businesses to be able to accept all forms of payment, their environments will need to be updated. This situation is predicted to worsen further during the coming years [1]. As financial institutions grapple given the challenging task of accurately and quickly differentiating in this study fraudulent and legitimate transactions, they acknowledge the essential role that artificial intelligence (AI) plays in this quest. AI powered fraud detection systems offer unparalleled speed, efficiency, and adaptability. The field of ML offers methods for developing models. ML has the potential to resolve the issues. Innovative ML techniques that can recognize increasingly intricate patterns and huge quantities of data [3].

ML algorithms examine past transaction data to spot trends and abnormalities linked to fraud. The models are trained using extracted features, which include transaction amount, location, time, merchant, and customer behavior. In order to start the predictive modeling process, historical transaction data which contains information about every CC transaction, including the cardholder's name, amount, location, merchant, time of day, and more be gathered [4]. Traditional fraud detection systems, which rely on pre-defined criteria or fixed rules, often fail to detect complex and dynamic fraud patterns. Detection algorithms need to be extremely sensitive and

adaptable in real time because scammers are often changing their strategies. The dynamic and varying nature of the data is additionally an important challenge to CCF identification. Since fraudulent transactions represent just over one percent of total transactions, traditional algorithms unable to detect them efficiently. By using previous data to identify hidden trends that can point to fraud, ML algorithms offer a powerful substitute. These algorithms can reduce false and improve accuracy when properly trained and tuned. However, for ML-based fraud detection systems to be successful, significant problems such feature selection, data imbalance, and model interpretability must be fixed.This paper explores the application of various ML algorithms in the identification of CCF, with a focus on understanding their benefits, drawbacks, and performance in real-world scenarios.Challenges in Fraud Detection Highlight the evolving nature of fraud tactics and the complexity of detecting fraud in real-time .The paper's primary contributions include addressing the first challenge. And also looked for the best answer to it in this study because, in our, it is one of the most difficult problems, and in this study used ML algorithms because they have proven their efficiency in this type of problem. The Role of ML Introduce how ML is revolutionizing fraud detection . Transactions are conducted daily. Both customers and financial institutions are exposed to illegal activities that result in significant losses.

2 RELATED WORK

This paper presents a review of various fraud detection techniques. Abnormal activity in electronic payment transactions is detected using fraud detection techniques.

M. Z. Mizher and A. B. Nassif [5] The CNN approach and two machine learning algorithms - SVM and random forest - were used to present CCF detection models. The models were evaluated and compared using severely skewed real-world CC data, and the random forest model performed the best with an accuracy of 99.7%. CNN, on the other hand, achieved 93.5% accuracy.

D. Sehrawat and Y. Singh [6] employed GRU and LSTM neural networks in conjunction with an auto-encoder to identify CCF. By eliminating the class labels, the autoencoder in the suggested method was able to execute representation learning from the data. To identify fraud, the LSTM and GRU models were fed the output of the auto-encoder along with the class labels. 99.1% classification accuracy was achieved by the LSTM.

J. Forough and S. Momtazi [7] A CCF detection model that takes into account the sequential nature of CC transactions was created. In an ensemble implementation, the technique employed LSTM models as basis classifiers with a feed-forward neural network (FFNN) serving as the voting mechanism. When tested on two CC datasets, the suggested LSTM ensemble performed better than alternative ML and DL methods. In particular, using the Brazilian and European datasets, the suggested ensemble had an AUC of 0.88 and 0.879.

I. D. Mienye and N. Jere [8] A model with MaxPooling layers that was built on BiLSTM and BiGRU was created. Three resampling methods were used to preprocess the dataset in the interim: SMOTE, random undersampling, and random oversampling. The deep learning-based classifier's performance was compared to that of various machine learning classifiers, such as logistic regression, random forest, voting, naïve base, AdaBoost, and DT. Excellent performance was achieved by the suggested BiLSTM-BiGRU when random oversampling was used.

T. Berhane and et al. [9] A CNN model's performance was compared to that of various machine learning methods, such as XGBoost, SVM, random forest, KNN, logistic regression, and decision. the with a 91.4% AUC.Two convolutional layers using the ReLu activation function, one flattened layer, and one fully linked layer make up the CNN model. According to the trial findings, the CNN outperformed the other classifiers with a classification accuracy of 97.2%.

In this paper cited a set of publications published between 2021 and 2024 that included fraud detection techniques, algorithms, and card fraud detection methods.

3 PROPOSED METHODOLOGY

Figure 1 shows the sequence of steps used during identifying CCF utilizing ML techniques:

1) Begin collecting the accessible data, then upload the CC set of data.
2) Extend data pre-processing and validate numerous imbalances in the data by using one-class classifiers and the Matthews correlation coefficient.
3) Plot the entire dataset's correlation matrix.

4) Divide the data into subgroups for testing and training, with 20% designated for testing and 80% designated for training.
5) Make use of ML techniques and detecting systems.
6) Determine the sum of the evaluation's various measures.

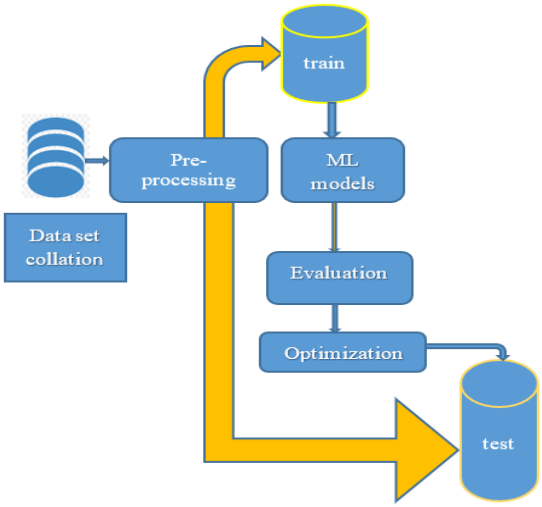

Figure 1: A proposed methodology for CCF using ML.

3.1 Data Set Collection

The CC Database is a dataset sourced from the website's Kaggle, with explanations available on the site [10] Several ML classifier techniques are implemented in this data analysis using the Python programming language to validate CCF from those in the dataset. That content for 1048575 CC, each card represent a row and each feature represent column. The columns are contents 10 features (type, amount, name, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, newbalanceDest, isFraud, isFlaggedFraud).

3.2 Exploratory Data Analysis (EDA)

EDA is a crucial step in the data analysis process. It involves analyzing datasets to identify their salient characteristics, sometimes with the use of visual aids. This is a summary of the normal procedures for carrying out EDA. Data summary from data frame and values. Number of rows 6362620 , Number of columns11. Data type from Column type and Count Float64 a count is 5, Int64 a count is 3, string account is 3. Table 1 provides summary statistics, specifically the mean and standard deviation (SD), for several features in a CC transaction dataset. This table highlights the variability in the financial features (balances and amounts), while also showing the rarity of fraud within the dataset. These statistics are essential for understanding the scale of the data and guiding the preprocessing steps in fraud detection tasks:

- Step. This variable represents the time step associated with each transaction. which has a mean value of 243.4 and a SD of 142.3. The comparatively high SD indicate that transaction times are dispersed over a wide range.
- Amount. Denotes the total amount of money exchanged, with an average of 179,900 and a SD of 603,900. The high SD indicates that there is a wide range in transaction amounts, with some involving significantly higher sums.
- Oldbalanceorg. This shows the sender's initial balance before to the transaction, with a mean of 833,900 and a SD of 2,888,000. The high SD suggests that there are significant differences in the originating accounts' balances, with some holding substantial sums of money.
- Newbalanceorig. With a mean of 855,100 and a SD of 2,924,000, this represents the sender's balance following the transaction. In many situations, the sender's balance may not change significantly following a transaction, as seen by the similarities between this and the previous feature (Oldbalanceorg).
- Oldbalancedest. This term describes the recipient's initial balance before to the transaction, which had an average of 1,101,000 and a SD of 3,399,000. This significant balance variance indicates that the beneficiaries' accounts also contain significantly different amounts.
- Newbalancedest. Shows the recipient's balance following the transaction; it has a mean of 1,225,000 and a SD of 3,674,000. The high SD indicates significant variance in post-transaction balances, which is consistent with the previously mentioned balance-related characteristics.
- IsFraud. This feature is binary and shows if a transaction is fraudulent or not. The dataset has a relatively low percentage of fraudulent transactions (mean of 0.001291, or roughly 0.13%), with a modest SD of 0.0359, suggesting that fraud cases are uncommon.
- Isflaggedfraud. This function shows if a transaction has been reported as possibly fraudulent. With a very small SD (0.001586) an an extraordinarily low mean (2.515e-06), it is

clear that very few transactions were reported for review.

Table 1: Exploratory data analysis.

Column_name	Mean	SD
Step	243.4	142.3
Amount	179900	603900
Oldbalanceorg	833900	2888000
Newbalanceorig	855100	2924000
Oldbalancedest	1101000	3399000
Newbalancedest	1225000	3674000
Isfraud	0.001291	0.0359
isflaggedfraud	2.515e-06	0.001586

3.3 Prepressing

Refers to the procedure of converting unstructured data into a clear, organized format that is appropriate for model training in the context of machine learning and data analysis. This stage is crucial since real-world data is often irregular and unformatted and may contain noise or errors that can negatively impact model performance. Figure 1 methodology contents refer into data representation dividing data into training dataset and test dataset through prepressing.

Steps in preprocessing:

1) Data Cleaning. Handling missing values use removing rows/columns with missing entries. And Removing duplicates Identifying and removing duplicate records.
2) Feature selection. selecting which features are most crucial in order to lower dimensionality and enhance model performance.
3) Splitting the Data. In this step, Two subsets of the data are divide: 20% for testing and 80% for training. The train_test_split function from sklearn makes this simple. Preprocessing is crucial because it improves model accuracy clean, well-preprocessed data helps ML models perform better. Properly splitting and preprocessing the data also guarantees that the model performs properly when applied to fresh, untested data. In this way, the model is trained on the training set and its performance is assessed on the test set.

Using methods for optimizing data scaling To enhance performance, four different data scaling techniques were used in this study to pre-process the data set and enhance the performance of ML models: StandardScaler, MinMaxScaler, RobustScaler, and PowerTransform, StandardScaler to scale data standardize features by eliminating the mean and scaling to unit variance, guaranteeing that the data follows to the Standard Normal distribution.

This helps ML algorithms and linear models perform more efficiently, especially when the data has different scales. MinMaxScaler That benefit of using data within a fixed range between 0 and .This is to preserve the relationship between data points and because this method does not need a normal distribution and because it is scaled between 0 and 1, especially for algorithms that are sensitive to feature size, such as neural networks

RobustScaler It was designed and used in this study to deal with outliers instead of using the mean and variance by scaling the data according to the interquartile range (IQR). This makes it more robust against outliers, ensuring that outliers do not distort the measurement process. PowerTransformer that was used and applied in this study because it makes the data more Gaussian-like. Models that assume normality in features, such as linear models, can improve model performance. Using these scaling techniques ensures that this study's ML models can effectively process the features, improving both accuracy and model generalization, particularly in cases where the features exhibit varying distributions or outliers.

3.4 ML Classifiers

This study used ML algorithms and predictive modelling to classify card-to-card transactions as fraudulent or non-fraudulent through the use of multiple cards, features, and preprocessing outcomes. The classification was performed using more than one classifier, all of which achieved high accuracy with a minimal number of errors. The classifiers consisted of DT, HistGradientBoosting, XGBoost, LightGBM, ANN, and CatBoost.

3.4.1 Decision Tree

DT algorithm that builds a tree model by utilizing fundamental tree ideas like node creation and branching, along with techniques like attribute selection and pruning. In the field of data mining [11].

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_{i.} \quad (1)$$

DT is used mostly for grouping purposes. Additionally, DT is a classification model that is frequently used in data mining. Each tree is made up of nodes and branches. Every subset specifies a value that the node can accept, and each node represents

features in a category that needs to be classified. Decision trees have found numerous application domains due to their straightforward analysis and accuracy on a variety of data formats[12].

3.4.2 Hist Gradient Boosting

The Hist Gradient Boosting algorithm builds on the foundation of Gradient Boosting. It modifies the traditional gradient boosting process by using histogram-based binning for efficiency, but the core mathematical principles remain similar. Here's a high-level view of the key equations used in gradient boosting, with additional notes on histogram-based optimization. Gradient boosting aims to minimize a loss function $L(y, \hat{y})$ over predictions $\hat{y}$ using an ensemble of decision trees. It iteratively improves predictions by adding a new tree in each iteration that focuses on the residuals (errors) of previous trees. Objective Function for Loss Minimization For N samples with features X and targets y, the goal is to minimize [14]:

$$\min_{\hat{f}\ (x)} \sum_{i=1}^{N} L\big(y_i, \hat{f}\ (x)\big). \qquad (2)$$

Hist Gradient Boosting divides feature values into distinct bins (histograms) rather than analyzing every data point when splitting nodes. Since only a small number of bins must be examined at each split, the method is sped up by mapping each feature value to one of these bins.

Gradient Boosting using residual errors (gradients) from earlier rounds, the technique iteratively adds weak learners (small decision trees) to enhance predictions. The gradient (or residual) of the prior trees is used to fit the weak learners. Regularization methods such as shrinkage (learning rate adjustment) can be used with Hist Gradient Boosting [13].

3.4.3 XGB

The gradient boosting decision tree (GBDT) is the foundation of XGB. The gradient boosting algorithm known as GBDT is DT-based. A simultaneous operation and an early stop are also supported by XGB. The tree may be stopped before the prediction result reaches its ideal value in order to expedite training [15]. Because it can handle big datasets and deliver state-of-the-art performance in a range of ML tasks, including classification, XGB is an ML approach that has become well-known and widely used [14]. A differentiable loss function $L(y, F(x))$, Input: training set $\{(x_\xi, y_i)\}_{i=1}^{N}$, Initialize model with a constant value [17]:

$$\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta). \quad (3)$$

Furthermore, because XGB used distributed and parallel computing, it operated faster. The XGB is often used for issues concerning ML and data mining due to its obvious benefits [15].

3.4.4 Light Gradient Boosting Machine

One gradient boosting system that makes use of decision trees as base learners is called Light Gradient Boosting Machine (LGBM). Large datasets and high-dimensional data are especially well-suited for it because of its scalable and efficient design. Particular refinements in its architecture have made LGBM renowned for its speed and performance, particularly in situations involving large volumes of data [14]. Its distinctive methods enhance its scalability and performance, yielding superior outcomes. LGBM builds trees leaf by leaf instead of going over every attribute in search of the optimal split. This allows it to perform a wide range of ML tasks with high accuracy and scalability [16] . LGBT is an additional gradient boosting algorithm that grows trees vertically using a leaf-wise process.To split and grow the tree, the leaf that minimizes the loss the greatest is selected.To find the optimal split possibilities, LGBM use a histogram-based approach. Gradient-based One-Side Sampling (GOSS) is a sampling algorithm used by LGBM to enhance training by highlighting the significance of data instances. Its primary purpose is to ignore data with tiny gradients and focus on data samples with bigger gradients [17]. Training Data $D = \{(\chi 1, y1), (\chi 2, y2), \ldots, (xN, yN)\}, \chi i \in \chi, \chi \subseteq R$, $yi \in \{-1, +1\}$;

Combine features that are mutually exclusive of $\chi i, i = \{1, \ldots, N\}$ by the exclusive feature bundling (EFB) technique [19]:

$$\text{Set } \theta_0(\chi) = \arg\ \min_c \textstyle\sum_i^N L(y_i, c). \quad (4)$$

3.4.5 Cat Boost Classifier

To improve its performance, usability, and automatic processing of categorical data, the CatBoost classifier algorithm is employed as a machine language tool to train datasets for fault classification. CatBoost additionally adds to the program's enhanced robustness by training on all sample datasets in the

algorithm. Prior to the sample, each sample's goal value is established. Weight and priority are then applied while the sample's properties are altered.With minimal data preprocessing, the CatBoost classifier can handle missing values for both numerical and non-encoded categorical variables [18]. Assuming a data sample size D = {(Xj, }Yi); j = 1, ...m,Where Xj = (x1j, x2j, ...xnj) is a vector of n features and response feature Yi 2R, which are binary and a sample (Xj,Yi) identically and independently distributed by an unknown distribution P(., .). The aim is to train a function H : Rn→ R that minimises the expected loss as shown in the following (5):

$$L(H) = EL(y, H(x)), \quad (5)$$

where L(., .) is a smooth loss function and (X, y) is a sample of test data drawn from the training data D [19].W_f is the feature weight ,P_f is the per-feature penalty,EP_f is the per-object penalty,S is the current split, L is the current leaf

$$U(f) = \begin{cases} 0, & \text{if } f \text{ was used in model already} \\ 1, & \text{otherwise} \end{cases}$$

$$U(f,x) = \begin{cases} 0, & \text{if } f \text{ was used already for object } x \\ 1, & \text{otherwise} \end{cases}$$

The final score is calculated as follows:

$$\text{Score}' = \text{Score} \cdot \prod_{f \in S} W_f - \sum_{f \in S} P_f \cdot U(f) - \quad (6)$$
$$- \sum_{f \in S} \sum_{x \in L} EP_f \cdot U(f,x) \text{ [21].}$$

3.4.6 ANN

This deep learning model [20] uses binary classification to identify fraudulent transactions in CC data. The model is constructed using PyTorch and employs a standard FFNN architecture. This structure consists of an input layer, two hidden layers, and an output layer. The input layer accepts feature vectors with a size equal to the number of preprocessed features using a power transformer The first hidden layer transfers the input to 64 neurons, followed by a ReLU activation function. The second hidden layer reduces this to 32 neurons using ReLU again. One neuron with a function of sigmoid activation that may produce probabilities ranging from zero to one that are accepted for binary classification makes up the final output layer.Binary Cross-Entropy Loss is the model's loss function, and it works well with binary targets. It was optimized using the optimization algorithm Adam with a rate of learning of 0.001.To ensure that each batch contains the correct sample composition, training data is loaded into the data loader in batches with shuffling enabled. The model goes through several stages. In this case 10 stages during training to compute predictions, calculate losses, adjust weights, and perform backpropagation. Accuracy and loss are calculated and verified at each stage during training and testing.

3.5 Evaluation

In this study, the confusion matrix, accuracy, and ROC curve were used to assess the ML model's performance for CCF identification. Below is the definition of each one separately.

3.5.1 Confusion matrix

One of the conventional methods for assessing the performance of ML models that yields four results. False Positive (FP), True Negative (TN), False Negative (FN), and True Positive (TP) are the four types. To assess the effectiveness of the classification model, this study obtains the confusion matrix, a matrix of NxN dimensions representing the values in the database. One axis of the matrix displays the average rating, and the other axis displays the actual data [21].

3.5.2 Accuracy

One way to determine how frequently a classifier correctly classifies a data piece is to look at its accuracy. According to the equation, the accuracy is determined by dividing the number of correctly classified cases (fraud (TP) and non-fraud (TN)) by the total number of accurate predictions (including TP and TN in binary classification) within the collection of occurrences [11].

3.5.3 ROC Curve

ROC Curve an analytical technique that uses a graph to show how well a binary diagnostic classification algorithm performs. Before the diagnostic test results can be placed into one of the well-defined dichotomous categories, a reference value (cut-off value) for diagnosis must be established because many test results are given as continuous or ordinal variables. To differentiate between noise (false positive result) and a signal (real positive result) [22].

4 EXPERIMENTAL RESULTS

The proposed methodology was implemented in Python. Software package running on Lenovo core i5, 8 GB RAM, 256 GB HD, AND Windows10 (64 bit) .for our experiments, these algorithms were applied to

real data from people's credit cards from the website. Several ML classifier techniques are implemented in this data analysis using the Python programming language to validate CCF from those in the dataset.

Figure 2 shows the two confusion matrices of the best classifier: one for the training set (left) and one for the test set (right). For a binary classification task, each matrix contrasts the predicted labels (columns) with the true labels (rows), where class 0 denotes the negative class and class 1 the positive class.

There are 5,338 real positives (properly predicted as class 1) and 4,120,804 true negatives (properly predicted as class 0) in the training set confusion matrix. The training data shows perfect recall for class 1 with 9,561 false positives (cases that were misclassified as class 1 and no false negatives (class 1 cases that were misclassified as class 0).

There are 2,844 true positives and 2,218,917 true negatives in the test set confusion matrix. 31 false negatives and 5,125 false positives have been detected. This indicates that the model performs well in both classes, as it achieves a lower misclassification rate than the training set, albeit with a slightly higher rate. The results were as shown in the Table 2. Before parallelizing or vectoring a program, scalar optimization is the process of improving its performance by removing inefficiencies from its original implementation. Code must be optimized for a certain execution environment in order to achieve best performance [22].

Table 2 shows a summary of the results reached by the paper, in which six classifiers were used with the data. In essence, the data was divided into two sets: the training set and the test set.

In this study found, while working on this data, that the highest results appeared for the Decisian Tree training set, where the result was without any error, while the XGBC was 0.9993. CatBoost was 0.9991, while HistGradientBoosting achieved the lowest results of 0.9974. Thus, the best result achieved in training was Decisian Tree 1.0000, while the lowest result was HistGradientBoosting 0.9974. The data was tested on six classifiers and the results appeared as follows: CatBoost 0.9966 is the highest result among all the classifiers, LGBM 0.9956 and HistGradientBoosting 0.9947, while the lowest results obtained were DT 0.9351. Thus, the CatBoost method is the best of all previous methods with a lower error rate and more efficient performance.

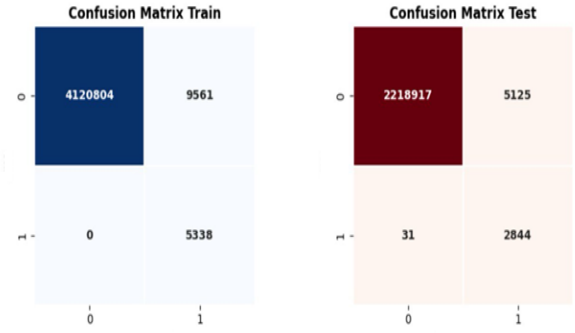

Figure 2: Results for the confusion matrix dataset.

Table 2: Results of ML with other classifiers.

Method	Train	Test
Hist Gradient Boosting	0.9974	0.9947
DT	1.0000	0.9351
CatBoost	0.9991	0.9966
LGBM	0.9981	0.9956
XGB	0.9993	0.9926
ANN	0.9996	0.8607

Figure 3 shows the training and test datasets' Receiver Operating Characteristic (ROC) curves. To evaluate the model's performance across a range of threshold values, the ROC curve shows the True Positive Rate (y-axis) versus the False Positive Rate (x-axis).

Both ROC curves for the training set (orange line) and the test set (blue line) are almost identical, running along the top left corner, indicating excellent classification performance. For both the training and test sets, the area under the curve (AUC) is displayed as 1.00, which represents perfect classification with no false positives or false negatives. This suggests that the model is highly effective, with no drop in performance between training and test datasets, although this could also indicate potential overfitting. The apple method on the Deep Learning ANN Balanced Accuracy Test: 0.8607.

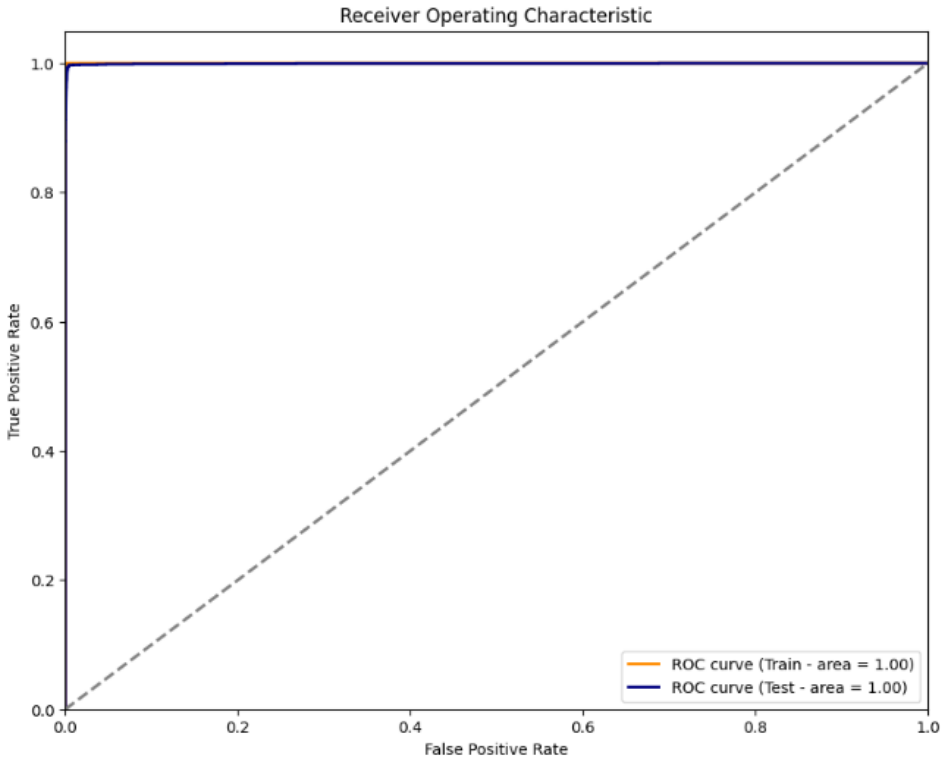

Figure 3: ROC curve of the CatBoost classifier.

5 CONCLUSIONS

A CC is considered among the most crucial things that must be available at the present time because people need it and it is used for most financial transactions such as salary, purchases from stores, paying water fees, paying electricity fees, and purchases from electronic store and etc.Credit cards are subject to many attacks aimed at fraud or theft of the amounts contained on the cards. The primary contributions of this paper are the creation of features for detecting payment card fraud and the utilization of actual transaction records to assess and illustrate the efficacy of the built. The data used in this study are cards that represent the number of rows, 6,362,620, and features which represent the number of columns 11. In this paper proposed a methodology to CCF detection through using by six ML algorithms CatBoost HistGradientBoosting, Decisian Tree, LGBM, ANN and XGBC. Comparing them with performance in terms of training and testing, divide the data into training and testing subgroups.80% as training as well as 20% also as testing .it was found that the best algorithm for training Decisian Tree classifier. And the best result for testing is CatBoost classifier these were the best results that were reached CatBoost 0.9966 is the highest result among all the classifiers. CCF detection was evaluated by confusion matrix, accuracy, and ROC curve. These algorithms were applied to real data from people's credit cards from the data website .It was suggested for Future work will apply the same steps used in this paper and apply them to deep learning algorithms and compare them.

REFERENCES

[1] F. K. Alarfaj, I. Malik, H. U. Khan, N. Almusallam, M. Ramzan, and M. Ahmed, “Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms,” IEEE Access, vol. 10, pp. 39700–39715, 2022, doi: 10.1109/ACCESS.2022.3166891.

[2] F. M. Talaat, A. Aljadani, M. Badawy, and M. Elhosseini, “Toward interpretable credit scoring: Integrating explainable artificial intelligence with deep learning for credit card default prediction,” Neural Computing and Applications, vol. 36, no. 9, pp. 4847–4865, 2024, doi: 10.1007/s00521-023-09232-2.

[3] G. Zioviris, K. Kolomvatsos, and G. Stamoulis, “An intelligent sequential fraud detection model based on deep learning,” The Journal of Supercomputing, vol. 80, no. 10, pp. 14824–14847, 2024, doi: 10.1007/s11227-024-06030-y.

[4] J. M. Akinbusola Olushola, "Fraud detection using machine learning," ScienceOpen, 2024, doi: 10.14293/PR2199.000647.v1.

[5] M. Z. Mizher and A. B. Nassif, "Deep CNN approach for unbalanced credit card fraud detection data," in Proc. 2023 Advances in Science and Engineering Technology Int. Conf. (ASET), Feb. 2023, pp. 1–7, doi: 10.1109/ASET56582.2023.10180615.

[6] D. Sehrawat and Y. Singh, "Auto-encoder and LSTM-based credit card fraud detection," SN Computer Science, vol. 4, no. 5, p. 557, Jul. 2023, doi: 10.1007/s42979-023-01977-w.

[7] J. Forough and S. Momtazi, "Ensemble of deep sequential models for credit card fraud detection," Applied Soft Computing, vol. 99, p. 106883, Feb. 2021, doi: 10.1016/j.asoc.2020.106883.

[8] I. D. Mienye and N. Jere, "Deep learning for credit card fraud detection: A review of algorithms, challenges, and solutions," IEEE Access, vol. 12, pp. 96893–96910, 2024, doi: 10.1109/ACCESS.2024.3426955.

[9] T. Berhane, T. Melese, A. Walelign, and A. Mohammed, "A hybrid convolutional neural network and support vector machine-based credit card fraud detection model," Mathematical Problems in Engineering, vol. 2023, Jan. 2023, doi: 10.1155/2023/8134627.

[10] R. Sharma, "Fraud detection dynamics: Financial transaction," Kaggle Dataset. [Online]. Available: https://www.kaggle.com/datasets/rohit265/fraud-detection-dynamics-financial-transaction/data

[11] B. Baykara and A. Uk, "Impact of evaluation methods on decision tree accuracy," 2015.

[12] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," Journal of Applied Science and Technology Trends, vol. 2, no. 1, pp. 20–28, 2021, doi: 10.38094/jastt20165.

[13] M. Ibrahim, H. AbdelRaouf, K. Amin, and N. Semary, "Keystroke dynamics based user authentication using histogram gradient boosting," International Journal of Computers and Information, Sep. 2022, doi: 10.21608/ijci.2022.155605.1081.

[14] M. Shehab, R. Taherdangkoo, and C. Butscher, "Towards reliable barrier systems: A constrained XGBoost model coupled with gray wolf optimization for maximum swelling pressure of bentonite," Computers and Geotechnics, vol. 168, p. 106132, 2024, doi: 10.1016/j.compgeo.2024.106132.

[15] T. Chen and C. Guestrin, "XGBoost," in Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, New York, NY, USA: ACM, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[16] A. Ahmad, A. A. Ahmad, Y. Lasheng, M. Kholief, and A. Garba, "Evaluation of convolutional neural network and gradient boosting methods for bug severity classification," Mar. 2024, doi: 10.13140/RG.2.2.26560.96002.

[17] F. Alzamzami, M. Hoda, and A. El Saddik, "Light gradient boosting machine for general sentiment classification on short texts: A comparative evaluation," IEEE Access, vol. 8, pp. 101840–101858, 2020, doi: 10.1109/ACCESS.2020.2997330.

[18] V. N. Ogar, S. Hussain, and K. A. A. Gamage, "Transmission line fault classification of multi-dataset using CatBoost classifier," Signals, vol. 3, no. 3, pp. 468–482, Jul. 2022, doi: 10.3390/signals3030027.

[19] M. Habeeb, "Enhanced Android malware detection through artificial neural networks technique," Mesopotamian Journal of CyberSecurity, vol. 5, 2025, doi: 10.58496/MJCS/2025/005.

[20] Ž. Đ. Vujovic, "Classification model evaluation metrics," International Journal of Advanced Computer Science and Applications, vol. 12, no. 6, 2021, doi: 10.14569/IJACSA.2021.0120670.

[21] W. P. Tanner and J. A. Swets, "A decision-making theory of visual detection," Psychological Review, vol. 61, no. 6, pp. 401–409, 1954, doi: 10.1037/h0058700.

[22] S. Li, High Performance Parallelism Pearls, 2015, doi: 10.1016/B978-0-12-803819-2.00013-6.