Volume 10, Issue 3, March – 2025 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1761

Credit Card Fraud Detection Comparing Multiple Supervised Learning Algorithms for Optimal Accuracy

Ayan Kumar Mahato[1]; Cezan Mendonca[2]; Harita Jasani[3]; Hariharan B[4]

[1,2,3]B. Tech 4th Year, [4]Assistant Professor,
[1,2,3,4]Department of CSE, SRMIST, Chennai, India

Publication Date: 2025/04/12

Abstract: Supervised machine learning algorithms are widely used for classification problems across various domains. However, selecting the best model requires a thorough evaluation of accuracy, robustness, and generalization ability. This research compares multiple supervised learning techniques using real- world datasets, focusing on evaluation metrics such as accuracy, sensitivity, specificity, and AUC-ROC. The study also considers the risk of overfitting, using cross- validation techniques to strengthen the conclusions. Results indicate that AdaBoost achieves near-perfect accuracy while Stochastic Gradient Descent (SGD) provides a balanced performance and generalisation, making their hybrid or combination a preferable choice for fraud detection.

Keywords: *Machine Learning, Accuracy, Classification, Generalisation.*

How to Cite: Ayan Kumar Mahato; Cezan Mendonca; Harita Jasani; Hariharan B. (2025). Credit Card Fraud Detection Comparing Multiple Supervised Learning Algorithms for Optimal Accuracy. *International Journal of Innovative Science and Research Technology*, 10(3), 2641-2653. https://doi.org/10.38124/ijisrt/25mar1761.

I. INTRODUCTION

The concept everyone relied on for getting access to banking services for a long time was simply the in-person services provided by Banking Organisations, until Citibank and Wells Forgo Bank introduced internet banking application in the United States of America in the year 1996. This was the beginning of a huge surge in online transactions, especially after the use of Credit Card services were adopted into online banking. All of a sudden, there was a huge market to entertain the use of online Card features such as payments. Loans, etc. across different types of platforms like e-commerce sites, Social networking, online banking, work from home, etc.

Now the online transaction space is more crowded than ever, with millions and even billions of online transactions taking place every single moment. With this increase in online transactions also comes the fraud transactions and scams that take place online to steal information and data and money from certain users or organisations as a whole. Online Cyber crime now more than ever is at an all time high, threat actors can access online data using methods like Cross Site Scripting to gain access to websites as other users using their session cookies, and using such actions, even gain administrative access. After all these actions, they also have services to mask their identities such as using VPN's(Virtual Private Networks). One way to find them out would be to monitor the transactions happening to and from that account. These transactions will have certain features and attributes where the values will vary by a noticeable margin compared to legitimate transactions. Using these differences in attributes and properties between legit and fraud transactions combined with the modern day power and versatility of AI and ML, we can create systems and models using Machine Learning Algorithms which can assist the existing systems in place for purposes of detection fraudulent transactions.

In the research we have undertaken, we are comparing 12 Supervised Learning Algorithms, the Supervised here signifies that the model will be trained using labeled data. We use a balanced dataset, meaning the number of fraud and legit transactions is the exact same. However during Predictive and training sessions it will be imbalanced. Some of the models are Stochastic Gradient Descent, DecisionTree, Logistic Regression, Random Forest, Support Vector Machine, Perceptron. All the mentioned algorithms/ their models will be compared across multiple deciding factors across multiple stages of comparisons to be able to find out the optimal Algorithm and its model respectively for the task of detecting fraud transactions from a set of Supervised learning Algorithms.

II. LITERATURE REVIEW

For this research, we have referred many informative papers and studies done on topics co-relating to our research such as the {1} investigation of performance of Naive Bayes, knn and LRM on highly skewed credit card fraud data, undertaking a hybrid technique of under and over sampling of data. Evaluations done using confusion matrix outputs. The next study proposes a {2}3-stage fraudulent card detection system which relies on a) Detection of invalid and fake credit from legitimate ones by using the Luhn algorithm for card number validation, b) dynamic verification of card expiry date, and c) a script code validation for Card Verification Value (CVV) or Card Verification Code (CVC) that compute the total number of digits which should be within the specified range. The next study proposes a model to {3} handle imbalanced data using XGBoost classifier to detect fraud transactions. The typical technique pre-determines the threshold value, resulting in inefficiency where several threshold values are computed and compared to identify the ideal value that provides an optimal outcome and high efficiency. The next study {4} compares 3 ML algorithms (i.e. Random Forest, Logistic regression, and AdaBoost) and compared the machine learning algorithms based on their Accuracy and Mathews Correlation Coefficient (MCC) Score. In these three algorithms, the Random Forest Algorithm achieved the best Accuracy and MCC score. The Streamlit framework is used to create the machine learning web application.

The next study {5} is a research investigating the application of advanced machine learning techniques to effectively detect fraudulent transactions. The findings highlight the need for resilient, scalable, and real-time mechanisms to combat evolving fraud strategies. The next study {6} examines the latest advances and application in the field of machine learning-based credit card fraud detection where four machine learning algorithms have been analyzed and compared on the basis of their accuracies. It is found out that Catboost algorithm works best to detect credit card fraud with an accuracy of 99.87 percentage. The dataset for credit card fraud detection was taken from kaggle. The next study {7} proposes a method to overcome the problem of Credit Card identification by combining Deep Learning with Machine Learning techniques. To reduce the number of false negatives, this study has performed data-matching trials with the implementation of Deep Learning Techniques. Utilizing the suggested strategy, it is possible to locate Credit Card Fraud (CCF) remotely from any location. The next study {8} discusses the problem of detecting a credit score includes modelling of past transactions for credit cards with the facts of those who have been revealed to fraud. The version is then used to delay whether new transactions are fraudulent or are now not new transactions. In this method, focus was put on the analysis and preprocessing of several anomaly detection algorithms and record sets, such as "neighbor outliers" and "forest zone isolation" algorithms, in PCA-converted credit card transaction statistics.

The next study {9} shows how algorithms like Logistic Regression and Random Forest were used in creating Fraud Fort, an advanced system designed to detect credit card fraud. The study illustrates the efficacy of integrating both models in Fraud Fort. The results indicate the combined advantages of logistic regression and random forest so that fraud detection system can become strong, eventually leading to a more secure and reliable economic ecosystem. The next study {10} depicts how popular supervised and unsupervised machine learning algorithms have been applied to detect credit card frauds in a highly imbalanced dataset. It was found that unsupervised machine learning algorithms can handle the skewness and give best classification results. The next study {11} is about how financial institutions aim to secure credit card transactions and allow their customers to use e-banking services safely and efficiently. To reach this goal, they try to develop more relevant fraud detection techniques that can identify more fraudulent transactions and decrease frauds. Defining the fundamental aspects of fraud detection, the current systems of fraud detection, the issues and challenges of frauds related to the banking sector, and the existing solutions based on machine learning techniques. The next study {12} implements six widely used machine learning techniques for credit card fraud detection. Their efficacy is analysed based on the parameters such as accuracy, precision, recall, specificity, misclassification, and F1 score. Results show that machine learning techniques are helpful for credit card fraud detection. We strongly recommend using multiple machine learning techniques for fraud detection before they occur or in the process of occurrence. The next study {13} performs a comparative experimental study to detect credit card frauds, as well as to tackle the imbalance classification problem by applying different machine learning algorithms for handling imbalanced datasets. The study shows that there is no need to process imbalance dataset by applying resampling techniques to measure the performance of our classifiers and it is sufficient to measure the performance through the three-performance measurements (Accuracy, Sensitivity, and Area Under Precision/Recall Curve (PRC) to prove the accuracy of the prediction of classification.

III. MODULE DESCRIPTION

This section provides an overview of the different components that make up the fraud detection system. Each module is responsible for a specific task, ensuring the pipeline runs smoothly from raw data input to fraud classification. Explained below are the different modules and their key purposes:

A. *Data Preparation and Preprocessing*

This is the starting or initial phase of the process. It ensures the below given processes:

- *Import Libraries:* Loads essential Python libraries such as NumPy, Pandas, Scikit-Learn, Matplotlib, and Seaborn for data handling, model training, and visualization.
- *Load Dataset:* Ensure correct dataset is loaded for all the algorithms and all their models are trained on the same

dataset. We make use of a balanced dataset with equal share of fraud and legit attribute. The dataset is in the form of a CSV file and contains over half a million rows of data, with many columns with sensitive data having PCA based data.

- *Preprocess Data:* Handle instances of missing data points with mean, median or mode replacement, do encoding and detecting outlier using methods like Z-score analysis.
- *Feature Scaling:* To ensure uniform scale, standardise or normalise data in a certain specific format across all the data. Aids in convergence in Machine Learning models.
- *Data Splitting:* Using splitting functions like train_test_split_data to split the dataset into training and testing data in a specified ration, in our case, we have used 80% of the dataset for training and 20% to test against the trained and fitted model.

B. *Model Initialisation and Training*

Once the correct libraries have been imported, dataset has been imported and processed, we move to the steps given below which explain the continuing workflow.

- *Initialise Model:* Selecting the required Machine Learning Algorithm to train the model and based on which model, defining parameters upon which model will have initial calibration and setting.
- *Timer:* Using the Time library in python to calculate time taken to fit specific model. This becomes a big factor that contributes to deciding ultimately which algorithm's model will be more efficient to train, test and fit.
- *Model Fitting:* Trains the specific algorithm's model on the provided training dataset. Some algorithms are equipped with hyper parameters and optimising functions and features like grid search for optimal n value (eg. knn), etc.

C. *Model Predictions and Evaluation*

The trained and tested model will now be evaluated based on it prediction capabilities and performance overview done via below given steps:

- *Predict on Training Data:* Generates trained model predictions on the training dataset. A crucial component in detection of overfitting.
- *Predict on Test Data:* Uses the trained model to generate predictions on the unseen testing data, provides first proper result of how well model has been trained.
- *Calculate Training Accuracy:* Computes the model's accuracy on training data.
- *Calculate Testing Accuracy:* Computes the model's accuracy on testing data in order to evaluate generalisation performance.

D. *Performance Metrics and Analysis*

The following steps provide results based on detailed evaluation using standard classification metrics—

- *Matrix and Plotting:* Generating confusion matrix and plotting heat map of the confusion matrix for easy interpretation.
- *ROC AUC Score and curve plotting:* Computing and plotting the Receiver Operating Characteristic curve and Area Under Curve score. This gives us an idea and understanding of the trade-off between sensitivity(recall) and specificity.
- *Determining Optimal Threshold and Prediction Recalculation:* Identifying the best classification threshold for improving model accuracy and adjusts predictions based on the optimal threshold and re-evaluation performance.
- *Calculating Optimal Test Accuracy:* Measures the new accuracy score after determining threshold optimisation.

E. *Data Insights and Visualisation*

The following steps involve he use of visual representations to analyse key data characteristics:

- *Plotting Distributions:* V isualising the distributions of transaction amounts, their id's, plotting correlation heat map to display feature relationships.
- *Organise Metrics:* Summarise all evaluation factors and their values for comparison or study or further analysis.

IV. DESIGN METHODOLOGY

Methodology is used to describe the step-by-step approach to how the system as a whole was made and designed. What all parts have had to come together to make the system work. We will understand the methodology of our research below:

A. *Extra Trees Classifier (Extremely Randomised Trees)*

An ensemble learning method based on Decision Trees. Unlike Random Forest, Extra Trees selects features randomly and splits at random thresholds instead of calculating the best split. Faster than Random Forest due to random splits. Good for handling imbalanced datasets like fraud detection. Mathematical Formulation:

- Given a dataset $D = \{X_i, Y_i\}$ with features X and labels, Y Extra Trees
- Selects a subset of features f randomly.
- Picks a random split value instead of best split θ.
- Constructs multiple decision trees.
- Prediction:

$$\hat{Y} = \frac{1}{T} \sum_{t=1}^{T} h_t(X)$$

Where $h_t(X)$s the output from the t^{th} decision tree, and T is the number of trees.

B. *Perceptron*

One of the earliest binary classifiers. It learns a linear decision boundary to separate fraud (1) and non-fraud (0) transactions. Uses Stochastic Gradient Descent (SGD) for weight updates. Mathematical Formulation:

- Given input. , $X = (x_1, x_2, \ldots, x_n)$
- Weights. $W = (w_1, w_2, \ldots, w_n)$ and bias. b

$$y = W \cdot X + b$$

- Activation function (step function):

$$f(y) = \begin{cases} 1, & \text{if } y \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

- Weight Update Rule:

$$W = W + \eta(Y - \hat{Y})X$$

Where. η is the learning rate.

C. *AdaBoost Classifier*

Boosting algorithm that combines multiple weak classifiers to form a strong classifier. Assigns weights to misclassified instances and re-trains on them. Mathematical Formulation:

- Given classifiers. $h_t(X)$ AdaBoost assigns a weight. α_t

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - e_t}{e_t} \right)$$

Where. e_t is the weighted classification error.

- Final prediction:

$$H(X) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(X) \right)$$

D. *Gaussian Naive Bayes*

A probabilistic classifier based on Bayes' theorem. Assumes independence among features and normal distribution. Mathematical formulation:

- Given features. $X = (x_1, x_2, \ldots, x_n)$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- Gaussian probability for a feature. x_i :

$$P(x_i|Y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- Classification: Assign to the class with the highest posterior probability.

E. *Stochastic Gradient Descent*

Optimization method for large datasets. Iteratively updates weights for each instance instead of the entire batch. Mathematical formulation:

- Given cost function. $J(W)$

$$J(W) = \sum_{i=1}^{N} \log(1 + e^{-Y_i W^T X_i})$$

- Gradient update rule:

$$W = W - \eta \nabla J(W)$$

Where. η is the learning rate.

F. *Multi-Layer Perceptron*

Neural Network with hidden layers. Uses backpropagation and activation functions like ReLU or Sigmoid. Mathematical formulation:

- Forward propagation:

$$A = f(WX + b)$$

- Error computation using **Cross-Entropy Loss**:

$$L = -\sum Y \log(\hat{Y})$$

- Backpropagation updates weights using gradient descent.

G. *XGBoost*

Gradient Boosting algorithm optimized for speed and accuracy. Uses tree pruning and regularization to prevent overfitting. Mathematical formulation:

- Boosted trees minimize loss:

$$L = \sum (Y - \hat{Y})^2 + \lambda ||W||$$

H. *Random Forest*

Ensemble of Decision Trees. Reduces variance compared to a single tree. Mathematical formulation:

- Predictions are aggregated across trees:

$$\hat{Y} = \frac{1}{T} \sum_{t=1}^{T} h_t(X)$$

I. *Decision Trees*

Recursive partitioning based on feature splits. Uses Gini Impurity or Entropy for splitting. Mathematical formulation:

- Gini Imputiry:

$$G = 1 - \sum P_i^2$$

- Entropy:

$$H = -\sum P_i \log_2 P_i$$

J. *K-Nearest Neighbours*

Classifies based on the majority of K nearest data points. Mathematical formulation:

- *Distance Metric:*

$$H = -\sum P_i \log_2 P_i$$

K. *Support Vector Machine*

Finds the optimal hyperplane that separates classes.

➢ *Mathematical Formulation:*

- Decision boundary:

$$f(X) = W^T X + b$$

- Objective:

$$\min ||W||^2$$

- Subject to:

$$Y_i(W^T X_i + b) \geq 1$$

L. *Logistic Regression*

Predicts probability using sigmoid function.

➢ *Mathematical formulation:*

- Sigmoid Function:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(WX+b)}}$$

M. *Environmental Setup and Requirements*

The interactive environment used must be suitable for coding, particularly for Machine Learning and Deep Learning. Coding language of choice is Python, due to its ease of use and as it is the standard ML coding language to code models. Using a proper environment, in our case we have used 3 which are Google Colab, Jupyter Notebook and Kaggle. Writing and executing code in these notebooks in ipynb format files provides us with good visualisation of results including graphs and curves, and also properly highlighted errors and quick ways to resolve them, given below are the tables comparing the properties of the 3 different interactive environment platforms or services we have used—

Table 1: Dataset Handling Comparison

Feature	Jupyter Notebook	Google Colab	Kaggle Notebooks
Dataset Access	Local files, databases, cloud (manual setup required).	Google Drive, cloud storage, or direct uploads.	Direct access to Kaggle datasets; easy integration.
File Storage	Local machine storage.	Temporary cloud storage (resets after session).	Persistent storage (within Kaggle).
Integration with Cloud Services	Manual setup for AWS/GCP.	Seamless integration with Google Drive.	Kaggle datasets and APIs easily accessible.

A factor that affects the scale to which a research can be done is based on resource availability and feasibility, the ttable given below addresses the cost comparison of the Interactive environments—

Table 2: Cost Considerations Comparison

Feature	Jupyter Notebook	Google Colab	Kaggle Notebooks
Free Access	Free, but dependent on local hardware.	Free with limited resources; Pro versions available.	Free with fair usage limits.
Pro/Paid Plans	No paid plans (hardware- dependent).	Colab Pro ($9.99/month), Colab Pro+ ($49.99/month) for better GPUs and longer runtimes.	No paid version; completely free.

The environment of choice must have acceptable levels of support for Libraries and Frameworks so that we can ensure that all of our algorithms are able to run and provide us with models.

Table 3: Framework Support Comparison

Feature	Jupyter Notebook	Google Colab	Kaggle Notebooks
Pre-installed Libraries	Requires manual installation (pip install).	Most ML libraries pre-installed.	ML/DL libraries pre-installed (TensorFlow, PyTorch, etc.).
Custom Libraries	Full control; can install any package.	Can install new libraries (!pip install).	Can install new libraries (!pip install).
TensorFlow/ PyTorch Support	Full support if installed.	Pre-installed; supports TPUs.	Pre-installed; optimized for Kaggle competitions.

Another major factor that will definitely produce different values for us based on which environment is chosen is the performance levels at which each environment operates, the comparison of performance capabilities is given below.

Table 4: Performance Capability Comparison

Feature	Jupyter Notebook	Google Colab	Kaggle Notebooks
Processing Power	Limited by local machine resources (CPU/ GPU/RAM).	Free tier provides cloud- based GPUs (T4, P100, V100 in Pro/Pro+).	Free cloud GPUs (T4, P100); limited time usage.
RAM Availability	Dependent on local hardware (8GB, 16GB, etc.).	Up to 12GB (Free), 24GB (Colab Pro), 32GB (Colab Pro+).	16GB RAM for free users.
Disk Storage	Uses local disk; constrained by storage capacity.	Limited to 107GB (temp storage, resets after session).	20GB disk storage, persists across sessions.
Runtime Limitations	No restrictions (local execution).	12-hour session limit (Free), longer in Pro.	9-hour session limit, but state persists across runs.
Internet Dependency	Not required; runs offline.	Required; cloud-based.	Required; cloud-based.

We primarily use and refer to the data and results we obtain from Jupyter Notebook because it caters to the needs of the research team. It is best for local execution, it has persistent environment, full control over dependencies and more power efficient on our systems.

V. RESULTS AND EVALUATION

We have done a 2 Stage analysis and evaluation of all the algorithms based on their output values for each Performance Evaluation Metric. In Stage 1, we simply compared all 12 algorithms based on their training, testing, and overall accuracies and their fit time. We managed to eliminate 7 algorithms from the running for multiple reason, such as unable to finish process-LRM, SVM, MLPC. We also eliminated based on overfitting- XGBoost, Decision Trees, Random Forest, Extra Trees Classifier. This left us with 5 remaining algorithms to compare in Stage 2 using more Metrics.

Table 5: Stage 1 Comparison of Algorithms

Algorithm	Fit Time (s)	Train %	Test %
ETC	21.21	1.0	0.999824
Perceptron	0.70	0.686103	0.685850
Adaboost	116.79	0.999635	0.999674
Naive Bayes	0.20	0.994491	0.994521
SGD	107.21	0.748404	0.747920
MLPC	148.51	0.998533	0.998355
XGB	2.01	1.0	0.999718
RandomForest	240.81	1.0	0.999672
Decision Tree	22.65	1.0	0.999586
KNN	0.12	0.999059	0.998497
SVM	-	-	-
LRM	-	-	-

Stage 2 is the phase where we compare the algorithms that did not get eliminated in the comparison of Stage 1. Here, we compare the performance of the algorithm's models across more sensitive and a bigger set of Evaluation Metrics.We must also consider Over-fitting for most of these Metrics.

A. Highly Crucial Metrics

Four basic metrics are used in evaluating the experiments, namely True positive (TPR), True Negative (TNR), False Positive (FPR) and False Negative (FNR) rates metric respectively.

$$TPR = \frac{TP}{P} \qquad FPR = \frac{FP}{N}$$

$$TNR = \frac{TN}{N} \qquad FNR = \frac{FN}{P}$$

The highly crucial metrics directly assess how well the model identifies fraud cases while handling class imbalance.

- Sensitivity:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

- Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- F1-score:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Matthews Correlation Coefficient:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

- ROC-AUC:

Table 6: Stage 2 High Crucial Metrics Comparison

Metric	SGD	Perceptron	ABC	Gaussian NB	KNN
Sensitivity	0.76206	0.559397	0.999596	0.989044	0.999947
Precision	0.758883	0.748764	0.999754	1.0	0.996427
F1 Score	0.760468	0.640374	0.999675	0.994492	0.998209
MCC	0.519938	0.38419	0.999349	0.989103	0.996418
ROC AUC	0.76006	0.790711	0.999968	0.999107	0.999002

B. Moderately Important Metrics Consists of:

- Balanced Classification Rate(BCR) which ensures the model performs well across both classes (legitimate & fraud) and also balances sensitivity and specificity.
- Cohen's Kappa measures agreement between predicted & actual fraud cases and accounts for chance agreement.
- Log Loss penalizes incorrect confident predictions. Helps in optimizing probabilistic models like Logistic Regression & Neural Networks.
- Average Precision (AP Score) summarizes precision-recall tradeoff at different thresholds. Useful for comparing models.
- Optimal Threshold, based on the type of experimental setup being made, can matter even more Cohen's Kappa, which is true in our case.

Table 7: Stage 2 Moderately Important Metrics Comparison

Metric	SGD	Perceptron	ABC	Gaussian NB	KNN
BCR	0.759967	0.685085	0.999675	0.994522	0.998206
Cohen's Kappa	0.519434	0.3717	0.999349	0.989044	0.996412
Log Loss	8.651668	-	0.300422	0.057248	0.007822
Average Precision	0.692371	0.807149	0.999926	0.999456	0.999085
Optimal Threshold	8.086462×10^{-33}	-1.196368×10^{10}	0.502309	0.321221	0.88
Sensitivity	0.76206	0.559397	0.999596	0.989044	0.999947
Specificity	0.757874	0.812303	0.999754	1.0	0.996465

The remainder of the metrics are not of utmost importance in the case of our experiment pertains to Credit Card Fraud Detection. We would now switch over to representing the usable levels of our algorithms in the form of graphs.

C. Less Important Metrics

These metrics are still useful and referred to but, in comparison to the other metrics, their significance diminishes greatly in the case of a Credit Card fraud detection system:

- Training Accuracy and Testing Accuracy can be misleading in an imbalanced dataset and overall not crucial.
- Optimal Test Accuracy is not compulsory either because it doesnt pertain specifically to our needs of fraud detection.

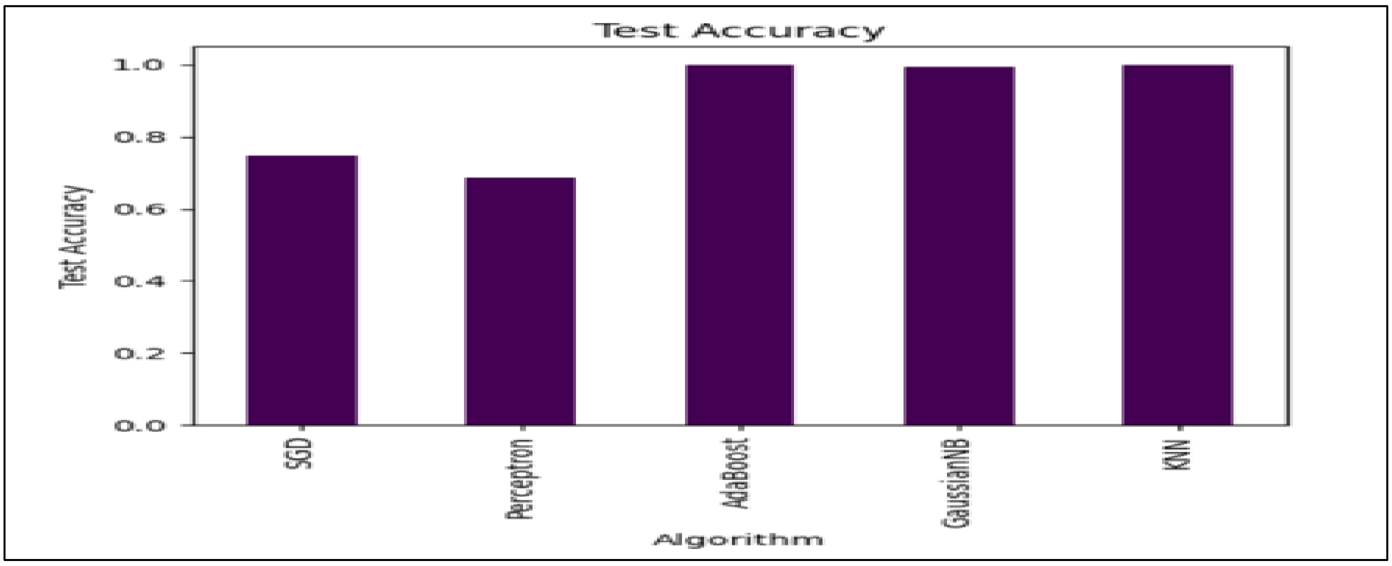

Fig 1: Graphical Comparison of Testing Accuracy

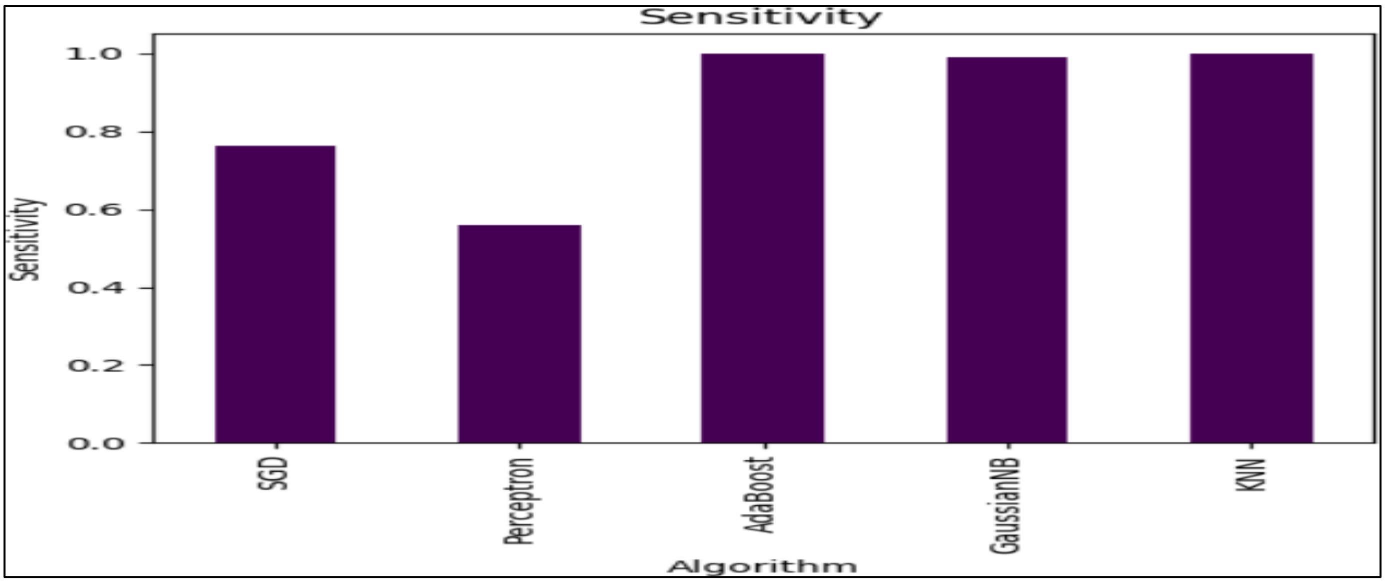

Fig 2: Graphical comparison of Sensitivity

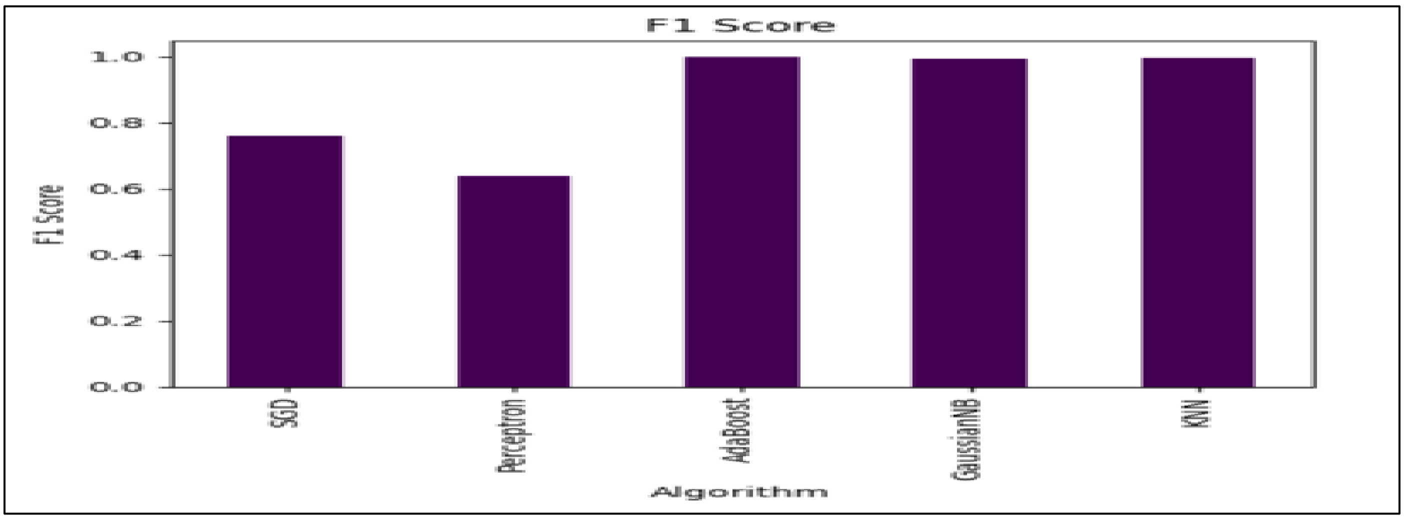

Fig 3: Graphical Comparison of F1-Score

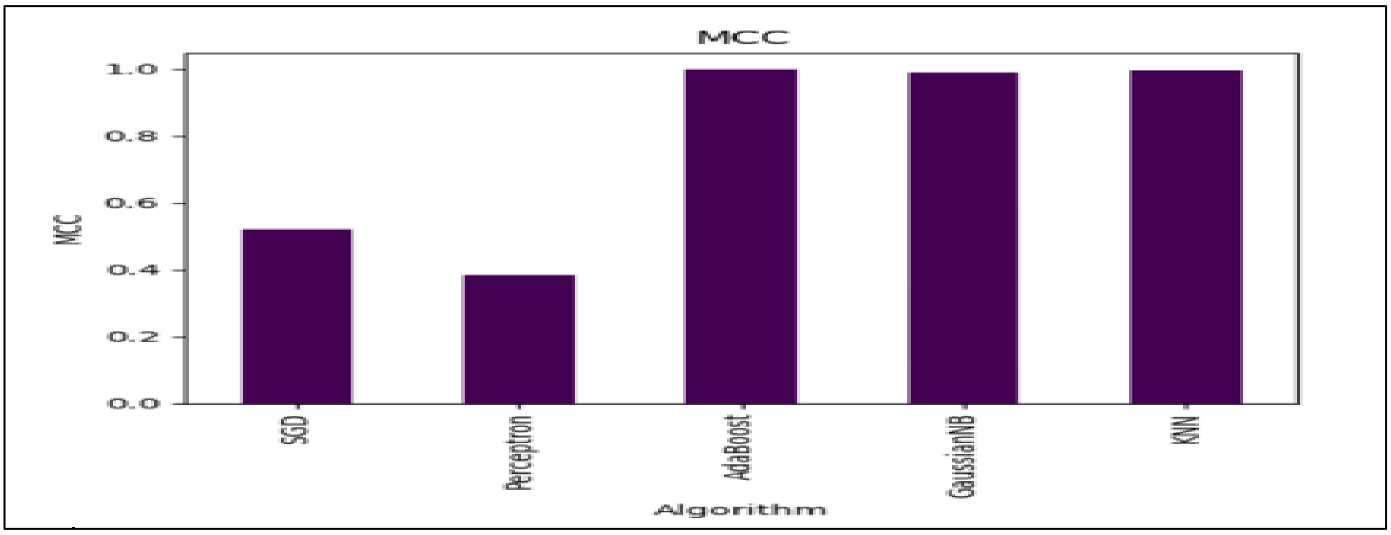

Fig 4: Graphical Comparison of MCC

VI. DISCUSSION

This section of our research allows us to discuss and come up with ideas on which algorithm or what combination of algorithms would be most suitable based on all the data we have collected and processed through and made the models to be tested for their predictive capabilities and fraud detection prowess. We will go over all of our findings point wise based on what category that factor falls into.

A. Best Practise for Model Selection

While we have narrowed down the set of algorithms from which we have to choose, we must also narrow down the criteria by which we choose the most optimal or most optimal combination of the algorithms and their models. For Credit Card Fraud detection, while Test Accuracy is not a high priority, we will still have to consider and eliminate all overfit models. That was the base criteria for Stage 1, for Stage 2 , the order of importance of criteria is given below—

Table 8: Ranking Importance of Crucial Metrics

Rank	**Metric**	**Importance**
1	Recall (Sensitivity)	Detects fraud cases (avoids false negatives)
2	Precision	Reduces false alarms (avoids blocking real transactions)
3	F1-Score	Balances Recall & Precision
4	ROC-AUC	Measures overall fraud detection ability
5	MCC	Best single-number metric for imbalanced data
6	BCR	Balances performance across both classes
7	Cohen's Kappa	Accounts for chance predictions
8	Log Loss	Useful for probabilistic models
9	Average Precision	Summarizes precision- recall tradeoff
10	Training/Testing Accuracy	Misleading in imbalanced datasets
11	Optimal Test Accuracy	Not directly useful for fraud detection
12	Specificity (TNR)	Less important than Recall & Precision
13	Training Time	Speed matters but not at the cost of fraud detection

B. Performance Evaluation Tables

We will now go through the results of each of our 5 algorithms individually discussing their values, how they stack up to our requirements based on Industry Ranges of the same values and also whether it is Optimal for Credit Card Fraud Detection. We will also discuss each of their Strengths and Weaknesses—

➢ *AdaBoost Classifier (Boosting)*

- *Strengths:*

✓ Excellent accuracy and generalization ability.
✓ Works well with imbalanced datasets (boosting improves minority class detection).
✓ Strong precision, recall, and F1-score.

- *Weaknesses:*

✓ Training time is longer due to boosting iterations.
✓ Sensitive to noisy data and outliers.
✓ May overfit on complex datasets.

Table 9: Ada Boost Performance Evaluation Table

Metric	Value	Industry Standard Compliance
Test Accuracy	0.9997	High (Excellent)
Precision	~0.9997	High
Sensitivity (Recall)	~0.9996	High
Log Loss	Low	Good (Indicates Confidence in Predictions)
F1 Score	~0.9996	High (Balanced Precision & Recall)
ROC AUC Score	~0.9997	High (Near Perfect Discrimination Ability)
MCC	High	Strong Positive Correlation

➢ *Perceptron (Linear Classifier)*

- *Strengths:*

✓ Computationally efficient (low training time).
✓ Works well in linearly separable problems.

- *Weaknesses:*

✓ Extremely poor performance in fraud detection.
✓ Cannot handle non-linearly separable data.
✓ Low precision, recall, and accuracy.

Table 10: Perceptron Performance Evaluation Table

Metric	Value	Industry Standard Compliance
Test Accuracy	0.4983	Low (Below Industry Standard)
Precision	~0.498	Low
Sensitivity (Recall)	~0.498	Low
Log Loss	High	Bad (Indicates Poor Confidence in Predictions)
F1 Score	~0.498	Low (Poor Balance of Precision & Recall)
ROC AUC Score	~0.50	Random Guessing Level
MCC	~0.0	No Correlation

➢ *Gaussian Naïve Bayes (NB)*

- *Strengths:*

✓ Fast & scalable (low training time).
✓ Performs well with independent features.

- *Weaknesses:*

✓ Assumes feature independence.
✓ Lower accuracy.

Table 11: Gaussian Naive Bayes (NB) Performance Evaluation Table

Metric	Value	Industry Standard Compliance
Test Accuracy	0.9945	High (Good)
Precision	~0.9944	High
Sensitivity (Recall)	~0.9945	High
Log Loss	Low	Good
F1 Score	~0.9944	High (Good Balance)
ROC AUC Score	~0.9945	High
MCC	High	Strong Correlation

➢ *K-Nearest Neighbors (KNN)*

- *Strengths:*

✓ Simple and effective model.
✓ Works well for small to medium datasets.
✓ Good performance across all metrics.

- *Weaknesses:*

✓ Computationally expensive for large datasets.
✓ **Memory-intensive** (stores all training data).
✓ Sensitive to choice of k and feature scaling.

Table 12: K-Nearest Neighbour (KNN) Performance Evaluation Table

Metric	Value	Industry Standard Compliance
Test Accuracy	0.9990	High (Good Enough)
Precision	~0.9985	High
Sensitivity (Recall)	~0.9984	High
Log Loss	Low	Good

F1 Score	~0.9985	High
ROC AUC Score	~0.9986	High
MCC	High	Strong Correlation

➢ *Stochastic Gradient Descent (SGD)*

- *Strengths:*

✓ Efficient for large datasets.
✓ Works well for high-dimensional data.
✓ Fast training time.

- *Weaknesses:*

✓ Low accuracy, precision, and recall.
✓ Does not generalize well to complex fraud detection patterns.
✓ Highly sensitive to learning rate and hyper parameters.

With the results we have tallied for each of the Algorithm's models, we can come up with a table that perfectly sums up the best points and the impending drawback of the algorithm with respect to our research on which would be ideal for a Credit Card Fraud Detection system. We will discuss alternative ways alongside picking the best one for the job and catering to industry metrics and values, meaning we must ensure that we not follow the biggest most pleasing values but ones that cater to the industry standard ranges for what the individual values must be.

Table 13: Stochastic Gradient Descent (SGD) Performance Evaluation Table

Metric	Value	Industry Standard Compliance
Test Accuracy	0.7479	Below Standard (Too Low)
Precision	~0.748	Low
Sensitivity (Recall)	~0.747	Low
Log Loss	Very Low (8.08e-303)	Good
F1 Score	~0.747	Low
ROC AUC Score	~0.748	Low
MCC	Low	Weak Correlation

Table 14: Best Use Case for Model Table

Algorithm	Best Use Case
AdaBoost	Excellent accuracy, works well with imbalanced data
KNN	Good accuracy but computationally expensive
Naive Bayes	Fast and scalable but makes independence assumptions
SGD	Fast with good Generalisation
Perceptron	Performs worse than random guessing

Combining SGD and AdaBoost is a promising approach because it balances generalization (SGD) with high accuracy (AdaBoost). This type of hybrid model can leverage the strengths of both algorithms:

- *SGD*: Works well with large datasets, avoids overfitting, and complies with industry standards.
- *AdaBoost*: Offers high accuracy, strong recall, and precision for detecting fraudulent cases.

Now we shall further discuss how to approach this idea of combining two algorithms and their use cases.

C. Combining SGD with AdaBoost

Now that we have established SGD gives us the most stable values and a very small margin of error by considering it's immensely low Log Loss value in comparison to the other algorithm's present in this algorithm, we will choose it as a potential model to work on a Credit Card Fraud Detection System alongside a model that works with far more efficiency and better handle on imbalanced data, that is AdaBoost.

This approach ensure working security in a real life scenario where the industry acceptable fit values of SGD ensure reliability and no over-fitting combined with the high levels of precision, sensitivity and security of imbalanced data handling abilities of AdaBoost classifier.

Given below are some methods by which we can combine the 2 algorithm's to make the most of their strong points while having minimal problems in implementation complexity:

➢ *Boosting SGD as a Weak Learner:*

AdaBoost traditionally works with weak classifiers (like decision stumps), but we can use SGD as a weak learner. Since SGD is fast and performs well in high- dimensional spaces, we can apply AdaBoost to iteratively improve it:

- Step 1: Train an SGD model on the dataset.
- Step 2: Use AdaBoost to assign more weight to misclassified samples.
- Step 3: Boost multiple weak SGD classifiers into a

stronger ensemble model.

- Pros:
  - ✓ Retains SGD's generalization power, reducing overfitting.
  - ✓ Boosts performance where SGD alone struggles.
- Cons:
  - ✓ Training time is higher due to boosting multiple SGD classifiers.

➢ *Hybrid Stacking Model:*

Instead of AdaBoost, we can stack SGD and AdaBoost separately, then use a meta-classifier (like Logistic Regression or a simple Neural Network) to make final decisions:

- Model 1: Train an SGD classifier to capture generalization and prevent overfitting.
- Model 2: Train an AdaBoost classifier to maximize recall and precision.
- Meta-Classifier: Combine predictions from both models to make a final decision.
- Pros:
  - ✓ Balances bias and variance (SGD reduces overfitting, AdaBoost improves accuracy).
  - ✓ Industry compliance while boosting detection power.
- Cons:
  - ✓ More computationally expensive (training two models.

➢ *Dynamic Model Switching:*

Use SGD for general cases and AdaBoost for high-risk cases:

- Step 1: Train SGD on the entire dataset (general fraud detection).
- Step 2: Identify high-confidence fraud cases using AdaBoost.
- Step 3: If SGD is unsure, use AdaBoost as a fallback decision-maker.
- Pros:
  - ✓ Reduces computational cost compared to full ensemble learning.
  - ✓ Uses SGD's generalization while relying on AdaBoost only when necessary.
- Cons:
  - ✓ Requires a **threshold mechanism** to decide when to switch models

With the necessary data tallied and all the subsequent discussions made, we will now provide the final verdict in the conclusion section of this research.

VII. CONCLUSION

➢ *Combining SGD with AdaBoost is not only Possible but Strategically Beneficial! It Helps Balance:*

- SGD's industry compliance (no overfitting, strong generalization).
- AdaBoost's high accuracy, recall, and fraud detection capability.

Table 15: Conditional Approach Table

Condition	Approach
If computation is not an issue	Stacking (SGD + AdaBoost + Meta-Classifier)
If efficiency is required	Dynamic Switching (SGD for general, AdaBoost for high-risk cases)

FUTURE SCOPE

Combining SGD (Stochastic Gradient Descent) and AdaBoost (Adaptive Boosting) for credit card fraud detection creates a balanced and robust system, leveraging SGD's compliance with industry standards and AdaBoost's high accuracy in complex data scenarios. Below are the key future scopes for such a system:

A. *Improved Fraud Detection Efficiency*

- The hybrid model can adaptively learn from new fraud patterns while avoiding overfitting, ensuring long-term effectiveness.
- SGD prevents overfitting, keeping the model aligned with real-world data, while AdaBoost enhances feature selection and identifies subtle fraud patterns.

B. *Real-Time Fraud Detection with Adaptive Learning*

- Using online learning capabilities of SGD, the system can update itself with new fraud cases without retraining from scratch.
- AdaBoost's adaptive nature helps in improving the classification of difficult fraudulent transactions.

C. *Scalability for High-Volume Transactions*

- The hybrid model can be deployed in large-scale banking and financial institutions, handling millions of transactions efficiently.
- Can be optimized for cloud-based deployment, enabling fraud detection in distributed financial systems.

D. *Robust Against Evolving Fraud Tactics*

- Fraud patterns continuously evolve, making it essential for models to adapt dynamically.
- The combined approach ensures resilience to emerging fraud techniques, reducing false positives and false negatives.

REFERENCES

[1]. J. O. Awoyemi, A. O. Adetunmbi and S. A. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis," 2017 International Conference on Computing Networking and Informatics (ICCNI),

Lagos, Nigeria, 2017, pp. 1-9, doi: 10.1109/ICCNI.2017.8123782.

[2]. Singh, A. Singh, A. Aggarwal and A. Chauhan, "Design and Implementation of Different Machine Learning Algorithms for Credit Card Fraud Detection," 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 2022, pp. 1-6, doi: 10.1109/ICECCME55909.2022.9988588.

[3]. Alneyadi, H. Lamaazi, M. Alshamsi, M. Albaloushi, M. Alneyadi and N. Megrez, "Toward an Efficient Credit Card Fraud Detection," 2024 Arab ICT Conference (AICTC), Manama, B a h r a i n , 2 0 2 4 , p p . 7 3 - 7 8 , d o i : 1 0 . 1 1 0 9 / AICTC58357.2024.10735025.

[4]. S. Jain, N. Sharma and M. Kumar, "FraudFort: Harnessing Machine Learning for Credit Card Fraud Detection," 2024 First International Conference on Technological Innovations and Advance Computing (TIACOMP), Bali, Indonesia, 2024, pp. 41-46, doi: 10.1109/TIACOMP64125.2024.00017.

[5]. S. Nijwala, S. Maurya, M. P. Thapliyal and R. Verma, "Extreme Gradient Boost Classifier based Credit Card Fraud Detection Model," 2023 International Conference on Device Intelligence, Computing and Communication Technologies, (DICCT), Dehradun, India, 2023, pp. 500-504, doi: 10.1109/ DICCT56244.2023.10110188.

[6]. V. Jain, H. Kavitha and S. Mohana Kumar, "Credit Card Fraud Detection Web Application using Streamlit and Machine Learning," 2022 IEEE International Conference on Data Science and Information System (ICDSIS), Hassan, India, 2022, pp. 1-5, doi: 10.1109/ICDSIS55133.2022.9915901.

[7]. V. R. Sonwane, S. Zanje, S. Yenpure, Y. Gunjal, Y. Kulkarni and R. Yeole, "Advanced Machine Learning Techniques for Credit Card Fraud Detection: A Comprehensive Study," 2024 5th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2024, pp. 1978-1981, doi: 10.1109/ICOSEC61587.2024.10722667.

[8]. Y. Singh, K. Singh and V. Singh Chauhan, "Fraud Detection Techniques for Credit Card Transactions," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2022, pp. 821-824, doi: 10.1109/ ICIEM54221.2022.9853183.

[9]. P. Y. Prasad, A. S. Chowdary, C. Bavitha, E. Mounisha and C. Reethika, "A Comparison Study of Fraud Detection in Usage of Credit Cards using Machine Learning," 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2023, pp. 1204-1209, doi: 10.1109/ ICOEI56765.2023.10125838.

[10]. S. Mittal and S. Tyagi, "Performance Evaluation of Machine Learning Algorithms for Credit Card Fraud Detection," 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2019, pp. 320-324, doi: 10.1109/CONFLUENCE.2019.8776925.

[11]. N. Boutaher, A. Elomri, N. Abghour, K. Moussaid and M. Rida, "A Review of Credit Card Fraud Detection Using Machine Learning Techniques," 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), Marrakesh, Morocco, 2020, pp. 1-5, doi: 10.1109/CloudTech49835.2020.9365916.

[12]. Shah and A. Mehta, "Comparative Study of Machine Learning Based Classification Techniques for Credit Card Fraud Detection," 2021 International Conference on Data Analytics for Business and Industry (ICDABI), Sakheer, Bahrain, 2021, pp. 53-59, doi: 10.1109/ICDABI53623.2021.9655848.

[13]. T. Baabdullah, A. Alzahrani and D. B. Rawat, "On the Comparative Study of Prediction Accuracy for Credit Card Fraud Detection wWith Imbalanced Classifications," 2020 Spring Simulation Conference (SpringSim), Fairfax, VA, USA, 2020, pp. 1-12, doi: 10.22360/SpringSim.2020.CSE.004.