International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.1 (2025) pp. 616-641
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Hyperparameter Tuning of Random Forest using Social Group Optimization Algorithm for Credit Card Fraud Detection in Banking Data

Sudhirvarma Sagiraju[1], Jnyana Ranjan Mohanty[2], Anima Naik[3,*]

[1]Research Scholar, School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, India,
Email: 2181071@kiit.ac.in - **ORCID:** 0009-0002-6481-4831

[2]Professor, School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, India,
Email: jmohantyfca@kiit.ac.in - **ORCID:** 0000-0002-8762-3037

[3]Professor, Computer Science and Engineering, Raghu Engineering college, Visakhapatnam, India,
* **Corresponding Author Email:** anima.naik@raghuenggcollege.in - **ORCID:** 0000-0002-7808-5994

Article Info:

DOI: 10.22399/ijcesen.777
Received : 16 December 2024
Accepted : 29 January 2025

Keywords :

SGO, Random Forest, accuracy, hyperparameters,
Credit Card Fraud Detection,
European cardholders' imbalanced dataset.

Abstract:

As the adoption of credit cards continues to expand alongside advancements in e-commerce, the frequency and complexity of fraudulent activities have also grown, posing significant challenges for the financial sector. Detecting fraudulent transactions within highly imbalanced datasets remains a critical issue in ensuring secure banking operations. This study explores a robust approach RF_SGO to credit card fraud detection by combining pre-processing techniques such as Synthetic Minority Oversampling Technique (SMOTE) and class weight adjustment with Random Forest (RF) models optimized using the Social Group Optimization (SGO) algorithm. Additionally, the study utilizes Random Forest's feature importance mechanism to identify the most influential features contributing to fraud detection, enhancing interpretability and decision-making. Our methodology evaluates RF_SGO across three datasets: the original European cardholders' imbalanced dataset, a class-weight-adjusted dataset, and a SMOTE-enhanced dataset. Model performance is measured using key metrics, including Accuracy, Precision, Recall, F1-Score, and ROC-AUC. The RF_SGO model demonstrated superior performance, with the SMOTE-enhanced variant achieving the highest ROC-AUC (0.98) and Recall (0.88), effectively balancing sensitivity and specificity. The class-weighted RF_SGO achieved the highest Precision (0.96), making it ideal for minimizing false positives. Furthermore, the feature importance analysis identified key predictors of fraudulent behavior, providing actionable insights for financial institutions. Comparisons with traditional machine learning algorithms (e.g., Logistic Regression, Decision Trees, and SVM) and advanced models (e.g., XGBoost, CatBoost, and deep learning) highlight RF_SGO's ability to outperform in precision-recall trade-offs and overall classification effectiveness. This study underscores the significance of incorporating hyperparameter tuning, feature importance analysis, and data balancing strategies to improve fraud detection. The proposed RF_SGO framework offers a scalable and efficient solution for financial institutions to mitigate fraud, ensuring more reliable and secure transaction systems.

1. Introduction

Credit card fraud detection is a critical area of research in financial security, driven by the rapid growth of online transactions and the escalating sophistication of fraudulent activities. The rise of digital payment methods and global e-commerce platforms has made transactions more convenient but also more vulnerable to exploitation. Fraudulent activities not only lead to financial losses but also erode customer trust in financial systems. According to industry reports, global credit card fraud losses have reached unprecedented levels, prompting a need for advanced and scalable fraud prevention systems. Financial institutions are under immense pressure to balance security with customer experience, as overly stringent measures can hinder legitimate transactions. Thus, designing effective fraud detection systems has become a strategic priority [1-2]. The detection of fraudulent transactions poses unique challenges due to the imbalanced nature of credit card datasets, where legitimate transactions far outnumber fraudulent ones. This imbalance often causes traditional machine learning models to favor the majority class, leading to poor performance in detecting fraudulent transactions. Furthermore, fraud patterns evolve rapidly as attackers employ sophisticated techniques such as identity theft, skimming, and phishing to bypass existing security measures. Models must not only detect known fraud patterns but also generalize well to emerging, unseen behaviors. Another critical challenge is minimizing false positives, as incorrectly flagging legitimate transactions can disrupt the customer experience and lead to loss of business. This necessitates the development of fraud detection systems that achieve a fine balance between precision and recall.

Machine learning techniques have emerged as powerful tools for identifying anomalous patterns in transaction data, enabling real-time fraud detection and prevention. Unlike rule-based systems, machine learning models can automatically learn from data, adapting to new fraud patterns with minimal human intervention [3]. Supervised learning algorithms, such as logistic regression, support vector machines, and ensemble methods like Random Forest, have demonstrated significant promise in fraud detection tasks. Additionally, advanced deep learning architectures, such as neural networks and autoencoders, offer opportunities to model complex relationships in high-dimensional data [4]. Unsupervised methods and anomaly detection algorithms are also gaining traction, especially for identifying previously unseen fraud scenarios. Combining these techniques with domain-specific knowledge can enhance the robustness of fraud detection systems [5].

This paper investigates novel approaches to enhancing fraud detection accuracy, with a particular emphasis on addressing class imbalance and reducing false positives. The study focuses on employing advanced machine learning algorithms and evolutionary optimization techniques to develop scalable models suitable for real-world applications. We propose an efficient credit card fraud detection framework, evaluated on publicly available European cardholders dataset [6], incorporating machine learning algorithms such as Random Forest (RF) and optimization methods like Social Group Optimization (SGO)[7], alongside hyperparameter tuning strategies. The SGO algorithm offers several benefits for hyperparameter selection in machine learning models, making it an effective choice for this purpose. SGO possesses a good optimality-finding ability and is used in various fields [8-18].

An effective fraud detection system must accurately identify fraudulent transactions while maintaining high precision, ensuring customer trust in financial institutions and minimizing losses from incorrect detections. The primary contributions of this paper are summarized as follows:

- **Addressing Imbalanced Data**: To mitigate the class imbalance issue in credit card fraud datasets, we employ Synthetic Minority Oversampling Technique (SMOTE) and class_weight-tuning of hyperparameters as preprocessing steps.
- **Fraud Detection Algorithm**: We adopt Random Forest as the base machine learning algorithm for detecting fraudulent transactions.
- **Feature Importance Analysis**: Using Random Forest, we analyze feature importance to quantify the contribution of each feature to the model's predictions, enabling better interpretability and feature selection.
- **Optimization Using SGO**: We utilize the Social Group Optimization (SGO) algorithm to fine-tune the hyperparameters of the Random Forest model, improving its predictive performance.
- **Comprehensive Evaluation**: To validate the proposed approach, we conduct extensive experiments on publicly available real-world datasets. The performance of the model is evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. The results demonstrate that the proposed methods outperform existing baseline approaches in detecting credit card fraud effectively.

This study highlights the potential of combining machine learning algorithms with evolutionary optimization techniques to build robust and efficient fraud detection systems, ensuring practical applicability in financial domains.

The remainder of this paper is organized as follows: Section 2 reviews the existing literature on credit card fraud detection methods and their limitations. Section 3 discusses SGO algorithm Section 4 outlines the methodology employed in this research. Section 5 discussed in detail about our proposed RF_SGO model. Section 6 presents the results and compares the performance of the proposed method with existing approaches. Discusses the findings, and their implications. Finally, Section 7 concludes the paper, summarizing the contributions and future research.

2. Related works

In reference [19], the authors designed a credit card fraud detection system utilizing various machine learning techniques, such as Logistic Regression (LR), Decision Tree (DT), Support Vector Machine

(SVM), and Random Forest (RF). These models were tested on a dataset containing transaction data from European cardholders in 2013. Due to the highly imbalanced nature of the dataset, the ratio of non-fraudulent to fraudulent transactions posed a significant challenge. The performance of the models was assessed based on classification accuracy, with LR, DT, SVM, and RF achieving scores of 97.70%, 95.50%, 97.50%, and 98.60%, respectively. While these results were promising, the authors suggested incorporating advanced data preprocessing methods to potentially enhance classifier performance further.

Varmedja et al. [20] proposed a method for detecting credit card fraud using machine learning, applying a dataset obtained from Kaggle [6]. This dataset, comprising transactions recorded over two days from European cardholders, exhibited a significant class imbalance. To address this, the authors utilized the Synthetic Minority Oversampling Technique (SMOTE). The models tested included RF, Naïve Bayes (NB), and Multilayer Perceptron (MLP). Their findings highlighted RF as the most effective, achieving an accuracy of 99.96%, while NB and MLP scored 99.23% and 99.93%, respectively. The study concluded by suggesting that incorporating feature selection methods could further enhance the accuracy of other models.

In [21], Khatri et al. evaluated the performance of various ML techniques, including DT, k-Nearest Neighbors (KNN), LR, RF, and NB, for credit card fraud detection. Using a highly imbalanced dataset sourced from European transactions, the models were evaluated primarily on precision. The study reported precision values of 85.11%, 91.11%, 87.5%, 89.77%, and 6.52% for DT, KNN, LR, RF, and NB, respectively.

Awoyemi et al. [22] conducted a comparative study of ML methods applied to fraud detection using the European cardholder dataset. The researchers addressed class imbalance through a hybrid sampling approach. Models tested included NB, KNN, and LR, and their evaluation was conducted using accuracy as the primary metric. The study reported accuracy scores of 97.92%, 54.86%, and 97.69% for NB, LR, and KNN, respectively. The authors noted the potential for improved results by incorporating feature selection techniques.

In [23], the authors investigated the use of several machine learning approaches to tackle credit card fraud detection, applying the European cardholder dataset. To manage the imbalance in the dataset, they employed SMOTE. Models like DT, LR, and Isolation Forest (IF) were evaluated using accuracy, with scores of 97.08%, 97.18%, and 58.83%, respectively.

Manjeevan et al. [24] presented a fraud detection framework that leveraged Genetic Algorithms (GA) for feature selection and aggregation. The study evaluated multiple machine learning models to assess the effectiveness of their approach. Results indicated that GA-RF achieved an accuracy of 77.95%, GA-ANN reached 81.82%, and GA-DT attained 81.97%. Khalilia et al. [25] investigated Decision Tree (DT), Logistic Regression (LR), Support Vector Machine (SVM), and Local Outlier Factor (LOF) methodologies using the same dataset. The study reported accuracy values of 97.08% for DT, 97.18% for LR, 95.12% for SVM, and 99% for LOF, with LOF achieving a precision of only 5%. Rtayli et al. [26] explored the performance of SVM, LOF, and Isolation Forest (iForest) models. The results showed that iForest achieved an accuracy of 99% with a precision of 34%, while DT reached 99% accuracy but suffered from zero precision. Ileberi et al. [27] employed SVM, Random Forest (RF), and DT for fraud detection on the European cardholders dataset. The results showed that SVM achieved an accuracy of 97.50%, RF achieved 98.60%, while DT had a precision score of 85.11%. Khan et al. [28] evaluated multiple models, including DT, k-NN, LR, RF, and NB, emphasizing precision as a key metric. The results showed that DT, k-NN, LR, RF, and NB achieved precision scores of 85.11%, 91.11%, 87.5%, 89.77%, and 6.52%, respectively.

Agarwal et al. [29] utilized LOF, iForest, DT, and XGBoost, as well as an XGBoost variant with Random Oversampling, to address fraud detection. LOF achieved an accuracy of 99.60% with a precision of 5%, while iForest scored 99.70% accuracy with 34% precision. DT achieved 99.80% accuracy with zero precision, and XGBoost achieved an accuracy of 99.96% with precision, recall, and F1-score values of 97.73%, 82.69%, and 89.58%, respectively. The XGBoost with Random Oversampling variant performed similarly, with precision, recall, and F1-score values of 96.63%, 82.69%, and 89.12%.

Noviandy et al. [30] applied XGBoost with various data balancing techniques, such as SMOTE, SMOTE-Tomek, SMOTE-ENN, and ADASYN. XGBoost with SMOTE achieved 99.95% accuracy, 89.69% precision, 83.65% recall, and an F1-score of 86.57%. With SMOTE-Tomek, it achieved 99.95% accuracy, 87.00% precision, 83.65% recall, and an F1-score of 85.29%. Using SMOTE-ENN, the results were 99.95% accuracy, 86.27% precision, 84.62% recall, and an F1-score of 85.44%. The ADASYN variant achieved 99.94% accuracy, 85.29% precision, 83.65% recall, and an F1-score of 84.47%.

Sinap [31] compared DT, LR, k-NN, RF, XGBoost, NB, and SVM. The results showed DT achieved 96% accuracy, 94% precision, 95% recall, and an F1-score of 95%. LR achieved 95% accuracy, 97% precision, 87% recall, and an F1-score of 92%. k-NN scored 97% accuracy, 96% precision, 96% recall, and an F1-score of 96%. RF achieved 97% accuracy, 99% precision, 94% recall, and an F1-score of 96%. XGBoost scored 96% accuracy, 98% precision, 91% recall, and an F1-score of 94%. NB achieved 94% accuracy, 96% precision, 86% recall, and an F1-score of 96%. Lastly, SVM scored 95% accuracy, 98% precision, 88% recall, and an F1-score of 93%.
The analysis of the related works on credit card fraud detection for the European cardholder dataset reveals certain limitations: Many studies highlight the inherent challenge of imbalanced datasets, where fraudulent transactions are significantly fewer than non-fraudulent ones. Techniques like SMOTE, hybrid sampling, and ADASYN have been applied to balance the dataset, but they might not generalize well for highly skewed datasets. These methods may lead to overfitting by oversampling the minority class, which does not fully address the complexity of distinguishing between fraud and non-fraud. Most works, such as those employing Random Forest (RF) or Decision Trees (DT), rely on fixed or empirically determined hyperparameters. Suboptimal hyperparameters can lead to degraded model performance in terms of accuracy, precision, recall, and F1-score. A heavy reliance on accuracy as the primary evaluation metric was observed in many studies. Accuracy can be misleading in imbalanced datasets as it may favor the majority class. Insufficient focus on metrics like precision, recall, and F1-score, which better reflect the performance on the minority class (fraudulent transactions). Feature selection or feature optimization techniques were not consistently applied or integrated into the detection models, as seen in some studies employing Genetic Algorithms (GA). Suboptimal feature selection can lead to noise in the model, reducing prediction performance.
How RF_SGO Can Address These**:** RF_SGO dynamically optimizes key hyperparameters of the Random Forest model, such as the number of trees, max depth, and split criteria, using the SGO algorithm. This approach ensures the selection of near-optimal hyperparameters tailored to the specific dataset, improving model performance. RF_SGO incorporates the optimized RF model, which inherently handles class imbalance better than other algorithms due to its ensemble approach. It can also be paired with advanced sampling techniques to further enhance balance. This minimizes the risk of overfitting while improving recall and precision for the minority class. RF_SGO evaluates fitness using a composite function (e.g., 1-accuracy) and optimizes for metrics like recall, precision, and F1-score rather than solely accuracy. This ensures that the model performs well across all metrics, addressing fraud detection challenges effectively.While RF_SGO primarily focuses on hyperparameter tuning, it can be extended to include feature selection during optimization, eliminating irrelevant or noisy features. This leads to more efficient and interpretable models, improving prediction accuracy and robustness. 1. Social Group Optimization (SGO) algorithm
The SGO algorithm is a population-based metaheuristic optimization technique inspired by the social behavior and decision-making patterns of human groups. It was first proposed as a novel approach to solve complex optimization problems by mimicking the way individuals in a group interact, share information, and collectively find solutions to achieve common objectives.

Key Concepts and Mechanisms

1.Population Representation:
In SGO, a population of solutions represents individuals in a social group. Each individual corresponds to a candidate solution for the optimization problem, characterized by specific parameters and a fitness value.
2.Social Interaction Phases:
The algorithm divides the optimization process into distinct phases, simulating real-world social dynamics:
•Improving Phase: Individuals evaluate their current position and adjust based on personal experience or local optimization strategies. This phase allows for exploration of the search space.
•Acquiring Phase: Individuals share information with others in the group, promoting collaboration and mutual learning. This interaction often leads to convergence towards optimal regions of the solution space.
3.Learning and Adaptation:
SGO incorporates learning mechanisms where individuals adapt based on the influence of better-performing solutions within the group. This mechanism enhances the algorithm’s exploitation capability.
4.Balance of Exploration and Exploitation:
By combining individual introspection and group interaction, SGO strikes a balance between exploration (searching new areas of the solution space) and exploitation (refining existing promising solutions).
Advantages of SGO

•Simplicity: The algorithm is straightforward to implement with minimal parameter tuning requirements.
•Scalability: SGO can handle high-dimensional and complex optimization problems effectively.
•Adaptability: It is versatile and can be applied across diverse domains, from engineering design to machine learning.
•Global Search Capability: SGO reduces the likelihood of getting trapped in local optima due to its collaborative exploration strategies.
SGO in This Study
In this research, the SGO algorithm has been employed to optimize the hyperparameters of the Random Forest (RF) model for credit card fraud detection. By effectively fine-tuning the model, SGO enhances classification performance, ensuring a superior balance between precision, recall, and overall accuracy.
The adaptability and efficiency of SGO make it an excellent choice for solving optimization problems in highly imbalanced datasets, as demonstrated in the context of fraud detection. This reinforces its potential for widespread use in real-world applications requiring robust and scalable solutions.

Here is the detailed **algorithmic framework** for the SGO algorithm:

1. Initialization

- Define the optimization problem: min *F(x)* or max *F(x),* where x is the decision variable and *F(x)* is the fitness function.
- Initialize a population of *N* individuals (solutions), x_i for *i*=1,2,...,*N*, within the predefined search space.
- Randomly assign initial positions for all individuals and calculate their fitness values $F(x_i)$
- Set parameters like the maximum number of iterations (*maxIter*), population size (*N*), and other control parameters.

2. Iterative Process

Repeat the following steps until the stopping condition (e.g., reaching *maxIter*) is met:

2.1. Improving Phase

- Each individual evaluates its current position using its fitness value $F(x_i)$
- Update the individual's position using:
  $$x_i^{new} = x_i + \alpha . r_1 . (x_{best} - x_i), \quad (1)$$
  Where α is a self-introspection parameter. r_1 is a random number in [0, 1]. x_{best} is the best solution found so far.

This step allows individuals to exploit their knowledge of the best solution.

2.2. Acquiring Phase

- Each individual interacts with a randomly chosen group member x_i, encouraging exploration :
  $$x_i^{new} = \begin{cases} x_i + r_2 . (x_i - x_k) + r_3 . (x_{best} - x_i) & if\ f(x_i) < f(x_k \\ x_i + r_2 . (x_k - x_i) + r_3 . (x_{best} - x_i) & otherwise \end{cases} \quad (2)$$
  Where r_2 and r_3 are random numbers between 0 and 1, which help to introduce diversity and exploration in the search process This interaction enables the population to explore different regions of the search space.

2.3. Boundary Checking

Ensure that the updated positions of individuals do not exceed the defined search space
bounds. Adjust any out-of-bound solutions accordingly.

2.4. Fitness Evaluation

Compute the fitness value F(xi)*F*(*xi*) for each updated individual.

2.5. Update Best Solution

If any individual achieves a better fitness value than x_{best}, update x_{best}.

3. Termination

Stop the algorithm when the maximum number of iterations is reached or the solution
converges to an acceptable threshold.

4. Return the Optimal Solution

Output the best solution x_{best} and its corresponding fitness value *F(*x_{best}*)*.

Algorithm 1 Pseudocode of SGO

```
1. Initialize population size (N), maximum iterations (maxIter), and other parameters.
2. Generate initial population of solutions within the search space.
3. Evaluate fitness of each solution and find the initial best solution ($x_{best}$).
4. For iter = 1 to maxIter:
   a. Introspection Phase:
      For each individual i:
         Update position using equation (1)
   b. Interaction Phase:
      For each individual i:
         Interact with random individual j:
         Update position using equation (2)
   c. Boundary Checking:
      Ensure all solutions remain within the search space.
   d. Fitness Evaluation:
      Recalculate fitness values for updated positions.
   e. Update Best Solution:
      If a better solution is found, update $x_{best}$.
5. End For
```

6. Return x_{best} and $F(x_{best})$

3. Methodology

The proposed method follows a structured approach to identify an efficient and accurate algorithm for detecting credit card fraud, as illustrated in figure 1. Given the highly imbalanced nature of the credit card fraud dataset, we addressed this issue using two independent techniques: the Synthetic Minority Oversampling Technique (SMOTE) [32,33] and class weight tuning [34] as a hyperparameter adjustment. Additionally, we conducted experiments without these techniques to analyze their impact on fraud detection. To enhance model performance, we employed a Random Forest (RF) classifier and optimized its hyperparameters using the Social Group Optimization (SGO) metaheuristic algorithm. This optimization aimed to identify the optimal parameter set to improve the model's accuracy on imbalanced fraud data.

The methodology begins with data acquisition, followed by preprocessing, which includes three critical steps: data cleaning, balancing the dataset, and feature selection. We utilized RF's feature importance mechanism to identify the most significant features, ensuring that only essential attributes were retained for building the predictive model.

The pre-processed data was then used to train RF models designed to detect fraudulent transactions effectively. Before model training, the hyperparameters of RF were fine-tuned using the SGO algorithm to achieve an optimal configuration. The dataset was divided into 80% for training and 20% for testing using the train_test_split method. The model was trained on the training subset and subsequently evaluated on the test subset to measure its predictive accuracy. To comprehensively assess the model's performance, we generated a confusion matrix and calculated various evaluation metrics, including accuracy, precision, recall, F1 score, and ROC-AUC score. These metrics provided a thorough evaluation of the model's effectiveness in detecting credit card fraud.

3.1 Feature Importance using Random Forest

Feature importance in the Random Forest (RF) algorithm [35] is a technique used to quantify the contribution of individual features to the model's predictions. This metric provides valuable insights into the dataset, helping identify which features are most influential in making decisions within the RF model.

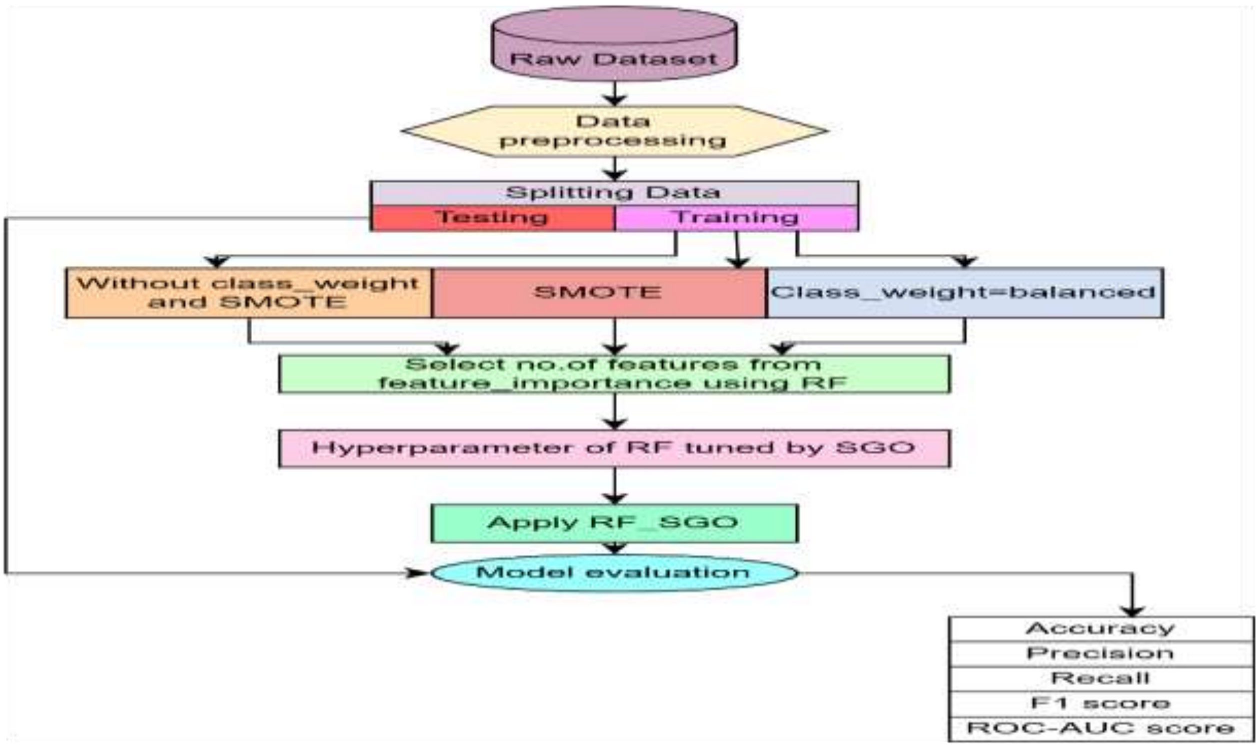

Figure 1. *The proposed architecture for credit card fraud detection*

It serves as an essential tool in enhancing model performance and interpretability.

Mechanism of Feature Importance
Random Forest is an ensemble learning method that builds multiple decision trees during training. Feature importance is calculated based on the extent to which a feature reduces the impurity (e.g., Gini impurity or entropy) across all trees in the forest. The importance score for each feature is derived from:

- **Impurity Reduction:** Evaluating how much the inclusion of a feature decreases impurity (improves the split quality) across all splits in all trees.
- **Permutation Importance:** Measuring the drop in model accuracy when a specific feature's values are randomly shuffled, thereby disrupting its relationship with the target variable.
- **Mean Decrease in Accuracy:** Estimating the reduction in accuracy if a feature is removed, aggregated across all trees.

Significance in Fraud Detection
In creditcard fraud detection, feature importance plays a critical role in identifying the most predictive attributes, such as transaction amounts, timestamps, locations, and user behaviour patterns. By focusing on these features, the model can be fine-tuned to enhance its precision and recall, ensuring more accurate detection of fraudulent transactions.

Advantages

- **Dimensionality Reduction:**

Feature importance helps reduce the number of input variables by retaining only the most significant ones, thereby simplifying the model and reducing the risk of overfitting.

- **Model Interpretability:**

It provides a clear understanding of how different features influence the model's predictions, aiding in decision-making and validating the model’s logic.

- **Optimized Feature Selection:**

It guides researchers in selecting the most relevant features, saving computational resources and improving model efficiency.

Implementation in this research
In this research, RF's feature importance was utilized to identify the critical features in the creditcard fraud detection dataset. This allowed us to prioritize influential features and exclude irrelevant ones during model training. The feature importance scores were computed using RF's built-in mechanism, and the results informed the preprocessing and model development phases.
By employing this technique, the proposed method achieved a refined feature set, enabling the RF classifier to deliver improved performance metrics in terms of accuracy, precision, recall, F1 score, and ROC-AUC. The insights derived from feature importance also contributed to the interpretability and reliability of the developed fraud detection system.

3.2 Performance metrics: Confusion matrix, and Evaluation Metrics

The confusion matrix[36] is a 2×2 matrix used to evaluate classification model performance, consisting of four elements: True Negative (TN), False Positive (FP), False Negative (FN), and True Positive (TP), arranged at positions (1,1), (1,2), (2,1), and (2,2), respectively. This matrix divides the predicted outcomes of a classification model into four categories:

- True Negative (TN): Correctly classified negative cases.
- False Positive (FP): Incorrectly classified positive cases.
- False Negative (FN): Incorrectly classified negative cases.
- True Positive (TP): Correctly classified positive cases.

.

Evaluation Metrics
Accuracy
Accuracy measures the proportion of all correct predictions relative to the total predictions made.

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

Precision
Precision, also known as positive predictive value, measures the proportion of predicted positive cases that are actually positive. It indicates the accuracy of positive predictions.

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

Recall
Recall, also referred to as sensitivity or true positive rate, represents the proportion of actual positive cases that are correctly predicted by the model.

$$\text{Recall(Sensitivity)}=\frac{\text{TP}}{\text{TP+FN}}$$

Specificity
Specificity measures the proportion of actual negative cases that are correctly identified by the model. It is also known as the true negative rate.

$$\text{Specificity}=\frac{TN}{TN+FP}$$

F1 Score

The F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both. It is especially useful when there is an uneven class distribution or when both precision and recall are important.

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

ROC-AUC Score: Evaluation Metric for Classification Models

The ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) score[37] is a comprehensive and widely recognized metric for evaluating the performance of classification models, particularly in scenarios involving imbalanced datasets, such as creditcard fraud detection. The ROC-AUC score reflects the model's ability to distinguish between the positive and negative classes effectively across different thresholds.

ROC Curve

The ROC curve is a graphical representation that illustrates the trade-off between two critical metrics:

- True Positive Rate (TPR) or Sensitivity:

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

This metric measures the proportion of actual positive cases correctly identified by the model.

- False Positive Rate (FPR):

$$\text{FPR}=\frac{FP}{FP+TN}$$

This metric quantifies the proportion of actual negative cases incorrectly classified as positive by the model.

AUC (Area Under the Curve)

The AUC value represents the area under the ROC curve and provides a single scalar measurement of the model's performance:

- An AUC value of 1 indicates perfect classification.
- An AUC value of 0.5 indicates performance equivalent to random guessing.

In this study, the ROC-AUC score was used as a key performance metric to evaluate the proposed creditcard fraud detection model. By summarizing the model's ability to distinguish between fraudulent and legitimate transactions across various thresholds, the ROC-AUC score ensured a comprehensive assessment of its predictive capabilities. The high ROC-AUC scores obtained in the experiments demonstrate the robustness and reliability of the model in identifying fraudulent transactions effectively.

4. Proposed RF_SGO model

To employ **SGO** algorithm for optimizing Random Forest (RF) model hyperparameters, it is essential to identify the most critical hyperparameters that influence RF's performance. The key hyperparameters considered in this study include:

- **Number of Trees (n_estimators)**: Defines the number of decision trees in the forest. Range: [10, 1000].
- **Maximum Depth of Trees (max_depth)**: Limits the depth of the tree to control overfitting. Range: [5, 50].
- **Minimum Samples Split (min_samples_split)**: Specifies the minimum number of samples required to split an internal node. Range: [2, 10].
- **Minimum Samples per Leaf (min_samples_leaf)**: Determines the minimum number of samples required to form a leaf node. Range: [1, maximum features].
- **Maximum Features (max_features)**: Indicates the number of features to consider for the best split. Range: [1, total features in the dataset].
- **Bootstrap Sampling (bootstrap)**: Indicates whether bootstrap samples are used when building trees. Values: **True** or **False**.
- **Criterion**: Specifies the function to measure the quality of a split. Values: **entropy** or **gini**.

In this study, each RF model is represented as a **decision vector** or **population** of seven dimensions (one per hyperparameter).

5. Experimental Design, Results Analysis and discussions

Machine learning library

The experiments on the ML techniques discussed in this study were conducted using Python, and the Scikit-learn library, commonly known as sklearn, which is a free package for machine learning [38]. This study also utilized various scientific computing libraries, including Scikit-learn [39], NumPy [40], matplotlib [41], pandas [42], and seaborn [43], to support the analysis and implementation.

Datasets

In this study, we utilize a real-world dataset to ensure the practical applicability of the proposed algorithm. The dataset, named "creditcard," comprises 284,807 transaction records collected over two days in September 2013. Of these, 492 transactions are identified as fraudulent, while the remainder are legitimate. The fraudulent transactions represent only 0.172% of the total, making this dataset highly imbalanced. This dataset is publicly accessible through Kaggle.

The dataset consists exclusively of numerical input variables derived from a Principal Component Analysis (PCA) transformation, as the original features and contextual details are unavailable due to confidentiality and privacy constraints. The PCA transformation produced components labelled V1 to

Algorithm to Decode a Decision Vector

The **decision vector** uses real-value encoding, which is decoded into discrete hyperparameter values during optimization. Algorithm 2 details the decoding process, as follows:

Algorithm 2: Decode a Decision Vector to RF Model

```
Input: Decision vector $V = [v_1, v_2, \dots\dots, v_7]$ where $d$=7.
Output: Optimized RF model.
Steps:
1. Map Bootstrap and Criterion:
        • For bootstrap, round $v_6$:
            • ROUND($v_6$)=0: Set True.
            • ROUND($v_6$)=1: Set False.
        • For criterion, round $v_7$:
            • ROUND($v_7$)=0: Set entropy.
            • ROUND($v_7$)=1: Set gini.
2. Define RF Model Parameters:
    Decode other hyperparameters as follows:
```

$$RF = RandomForestClassifier(\begin{cases} n\_estimators = ROUND(v_1) \\ \max\_depth = ROUND(v_2) \\ \min\_samples\_split = ROUND(v_3) \\ \min\_samples\_leaf = ROUND(v_4) \\ \max\_features = ROUND(v_5) \\ bootstrap = Decoded\ Bootstrap \\ criterion = Decoded\ Criterion \end{cases}$$

```
3. Evaluate Fitness:
    Train the RF model on the training dataset and evaluate its accuracy on the test dataset. Calculate the fitness value:
```

$$f(V) = 1 - Accuracy$$

Decoded values correspond to:
RF: *n_estimators*=747,*max_depth*=14,*min_samples_split*=2,*min_samples_leaf*=1,*max_features*=5,*bootstrap*="True",*criterion*="entropy".

SGO Procedure for RF Optimization

Algorithm 3: RF_SGO Model Optimization

```
Input: Dataset features ($X$), target labels ($y$).
Output: Optimized RF model with near-optimal hyperparameters.

Steps:
1. Initialization:
    • Define SGO parameters: population size ($P$), introspection parameter ($c$), decision vector dimension ($d$=7), maximum iterations ($max\_gen$), and bounds for each hyperparameter.
    • Split the dataset into training and testing sets.
    • Randomly initialize the population, with decision vectors sampled uniformly within the bounds.
2. Fitness Evaluation:
    • Decode each decision vector using Algorithm 2 to create an RF model.
    • Compute the fitness $f(V) = 1 - Accuracy$.
```

3. Global Best Search:

- Identify the decision vector with the minimum fitness value (best accuracy).

4. Iteration (Optimization Loop):

Repeat until termination criteria (e.g., *max_gen*) are met:

- Perform **Improving Phase:** Update each decision vector based on self-introspection and the global best solution.
- Perform **Acquiring Phase:** Interact with other vectors to explore the solution space.
- Evaluate new decision vectors and retain the better solutions.

5. Final Model:

- Select the best decision vector from the final population.
- Decode it into an optimized RF model using **Algorithm 2**.

Optimization Objective

The SGO algorithm aims to minimize the fitness function $f(V)$:

$\min_{V_i} f(V_i)$, subject to $L_j \leq V_{i,j} \leq U_j$

Where:

- L_j and U_j are the lower and upper bounds of the j-th hyperparameter.
- $f(V_i)$ is computed as 1−Accuracy.

This process iteratively enhances the performance of the RF model by fine-tuning its hyperparameters. Figure 2 presents a graphical abstract of the proposed **RF_SGO** framework.

Table 1. *The feature of creditcard fraud dataset that is used in this paper*

Features name	Description	Type	Resources
$V_1, V_2, V_3, \dots\dots, V_{28}$	Transaction feature after PCA transformation	Integer	https://www.kaggle.com/mlg-ulb/ creditcardfraud.
Time	Seconds elapsed between each Transaction	Integer	
Amount	Transaction value	Integer	
Class	Legitimate or Fraudulent	0 to 1	

Table 2 *The transaction label distribution in the "creditcard" dataset. This unbalanced data is expected in real-life datasets.*

Operation	**No of Transactions**	**No. of legitimate Transaction**	**No. of fraudulent Transactions**	**Legitimate (%)**	**Fraudulent (%)**
Before train_test_split:80% for training, 20% for testing					
Original	284,807	284,315	492	99.83%	0.17%
Dataset after SMOTE operation	483459	284315	199144	58.81%	41.19%
Dataset with class_weight operation	284,807	284,315	492	99.83%	0.17%
After train_test_split:80% for training, 20% for testing					
Original : Training	199364	199008	356	99.82%	0.18%
Original : Testing	85443	85307	136	99.84	0.16%
Dataset after SMOTE operation: Training	398016	199008	199008	50%	50%

Dataset after SMOTE operation: Testing	85443	85307	136	99.84	0.16%
Dataset with class_weight operation : Training	199364	199008	356	99.82%	0.18%
Dataset with class_weight operation : Testing	85443	85307	136	99.84	0.16%

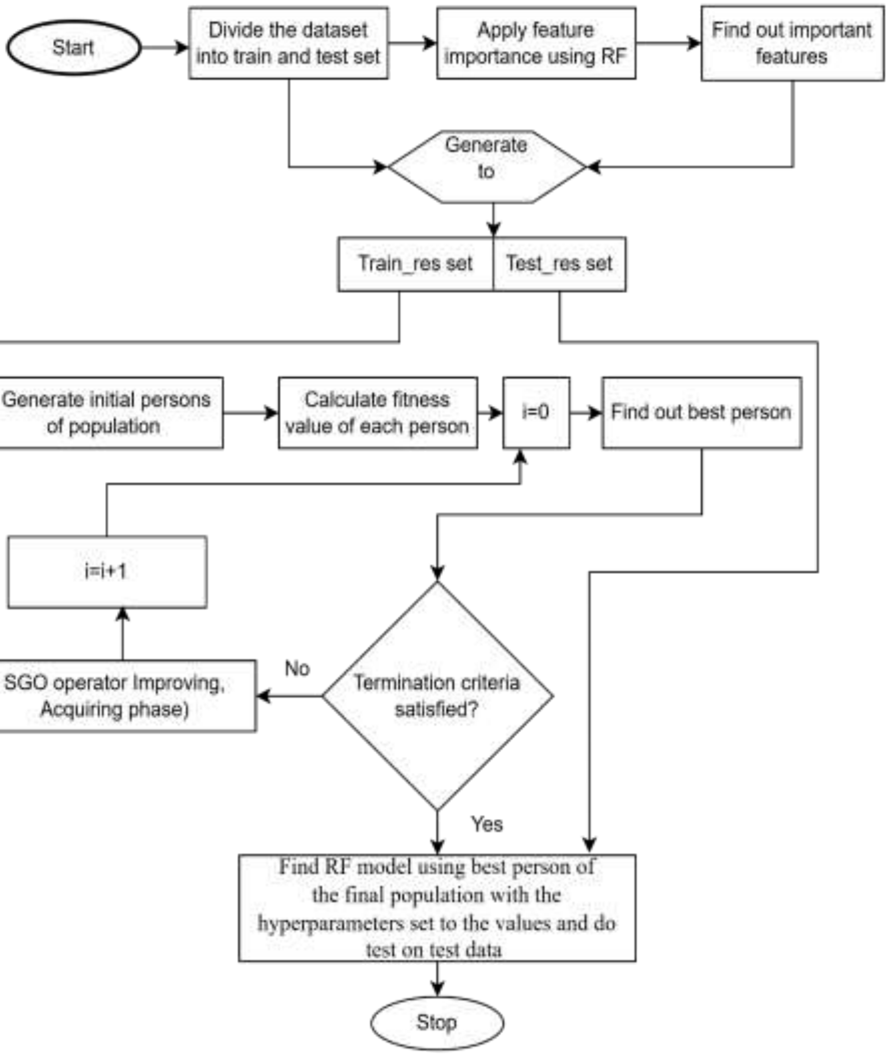
Start
Divide the dataset into train and test set
Apply feature importance using RF
Find out important features
Generate to
Train_res set
Test_res set
Generate initial persons of population
Calculate fitness value of each person
i=0
Find out best person
i=i+1
SGO operator Improving, Acquiring phase)
No
Termination criteria satisfied?
Yes
Find RF model using best person of the final population with the hyperparameters set to the values and do test on test data
Stop

Figure. 2 *Graphical abstract of proposed RF_SGO Model*

Table 3. *Performance Metrics of the Proposed Model with Numbers of Selected Features (5-17) for original credit card dataset*

No. of selected features	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Accuracy	0.999579	0.999555	0.999602	0.999602	0.999579	0.999625	0.999602	0.999637	0.999637	0.999649	0.999661	0.999625	0.999649
Precision	0.91	0.91	0.93	0.94	0.93	0.95	0.94	0.95	0.96	0.96	0.96	0.94	0.96
Recall	0.82	0.80	0.81	0.80	0.79	0.81	0.80	0.82	0.81	0.82	0.82	0.82	0.81
F1-score	0.86	0.85	0.87	0.87	0.86	0.87	0.87	0.88	0.88	0.88	0.89	0.87	0.88
ROC-AUC	0.94	0.94	0.95	0.95	0.95	0.96	0.95	0.96	0.96	0.95	0.95	0.96	0.96
Macro avg precision	0.95	0.95	0.97	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.98	0.97	0.98
Macro avg Recall	0.91	0.90	0.90	0.90	0.90	0.90	0.90	0.91	0.90	0.91	0.91	0.91	0.90
Macro avg F1-score	0.93	0.93	0.93	0.93	0.93	0.94	0.93	0.94	0.94	0.94	0.94	0.94	0.94
Weighted avg precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Weighted avg Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Weighted avg F1-score	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
TN	85296	85296	85299	85300	85299	85301	85300	85301	85302	85302	85302	85300	85303
FP	11	11	8	7	8	6	7	6	5	5	5	7	4
FN	25	27	26	27	28	26	27	25	26	25	24	25	26
TP	111	109	110	109	108	110	109	111	110	111	112	111	110

Table 4. *Performance Metrics of the Proposed Model with Numbers of Selected Features (8-30) for original credit card dataset*

No. of selected features	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Accuracy	0.999614	0.999649	0.999590	0.999625	0.999625	0.999649	0.999602	**0.999672**	0.999590	0.999614	0.999637	0.999625	0.999614
Precision	0.95	0.96	0.93	0.95	0.94	0.96	0.93	**0.96**	0.93	0.93	0.96	0.96	0.95
Recall	0.80	0.82	0.80	0.81	0.79	0.82	0.81	**0.83**	0.80	0.82	0.81	0.79	0.80
F1-score	0.87	0.88	0.86	0.87	0.86	0.88	0.87	**0.89**	0.86	0.87	0.88	0.87	0.87
ROC-AUC	0.95	0.95	0.96	0.96	0.96	0.96	0.96	**0.96**	0.96	0.96	0.97	0.96	
Macro avg	0.97	0.98	0.97	0.97	0.97	0.98	0.97	0.98	0.97	0.97	0.98	0.98	0.97

precision													
Macro avg Recall	0.90	0.91	0.90	0.90	0.90	0.91	0.90	0.92	0.90	0.91	0.90	0.90	0.90
Macro avg F1-score	0.93	0.94	0.93	0.94	0.93	0.94	0.93	0.94	0.93	0.94	0.94	0.94	0.93
Weighted avg precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Weighted avg Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Weighted avg F1-score	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
TN	85301	85302	85299	85301	85300	85302	85299	85302	85299	85299	85302	85303	85301
FP	6	5	8	6	7	5	8	5	8	8	5	4	6
FN	27	25	27	26	28	25	26	23	27	25	26	28	27
TP	109	111	109	110	108	111	110	113	109	111	110	108	109

Table 5 *Performance Metrics of the Proposed Model with Numbers of Selected Features (5-17) using SMOTE oversampling technique on credit card dataset*

No. of selected features	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Accuracy	0.997612	0.998514	0.998841	0.999204	0.999251	0.999309	0.999356	0.999391	0.999403	0.999462	0.999462	0.999520	0.999544
Precision	0.39	0.52	0.59	0.70	0.71	0.73	0.76	0.77	0.78	0.80	0.80	0.82	**0.84**
Recall	0.89	0.88	0.88	0.88	0.89	0.89	0.88	0.88	0.88	0.88	0.89	0.89	**0.88**
F1-score	0.54	0.65	0.71	0.78	0.79	0.80	0.81	0.82	0.82	0.84	0.84	0.86	**0.86**
ROC-AUC	0.97	0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.97	0.98	**0.98**
Macro avg precision	0.70	0.76	0.80	0.85	0.86	0.87	0.88	0.89	0.89	0.90	0.90	0.91	0.92
Macro avg Recall	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
Macro avg F1-score	0.77	0.83	0.85	0.89	0.90	0.90	0.91	0.91	0.91	0.92	0.92	0.93	0.93
Weighted avg precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Weighted avg Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Weighted avg	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

F1-score													
TN	85118	85196	85224	85255	85258	85263	85269	85272	85273	85278	85276	85281	85284
FP	189	111	83	52	49	44	38	35	34	29	31	26	23
FN	15	16	16	16	15	15	17	17	17	17	15	15	16
TP	121	120	120	120	121	121	119	119	119	119	121	121	120

Table 6. *Performance Metrics of the Proposed Model with Numbers of Selected Features (8-30) using SMOTE oversampling technique on credit card dataset*

No. of selected features	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Accuracy	0.999508	0.99954	0.999520	0.999508	0.999544	0.999508	0.999497	0.999497	0.999497	0.999497	0.999508	.999520	
Precision	0.82	0.84	0.83	0.82	0.84	0.83	0.83	0.83	0.82	0.82	0.83	0.84	
Recall	0.88	0.88	0.88	0.88	0.88	0.88	0.87	0.87	0.88	0.88	0.88	0.87	
F1-score	0.85	0.86	0.85	0.85	0.86	0.85	0.85	0.85	0.85	0.85	0.85	0.85	
ROC-AUC	0.97	0.97	0.97	0.98	0.97	0.97	0.98	0.97	0.98	0.97	0.98	0.97	
Macro avg precision	0.91	0.92	0.91	0.91	0.92	0.91	0.91	0.91	0.91	0.91	0.91	0.92	
Macro avg Recall	0.94	0.94	0.94	0.94	0.94	0.94	0.93	0.93	0.94	0.94	0.94	0.93	
Macro avg F1-score	0.93	0.93	0.93	0.93	0.93	0.92	0.92	0.92	0.92	0.92	0.92	0.93	
Weighted avg precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Weighted avg Recall	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Weighted avg F1-score	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TN	85281	85284	85282	85281	85285	85282	85282	85282	85281	85281	85282	85284	
FP	26	23	25	26	22	25	25	25	26	26	25	23	
FN	16	16	16	16	17	17	18	18	17	17	17	18	
TP	120	120	120	120	119	119	118	118	119	119	119	118	

Table 7. *Performance Metrics of the Proposed Model with Numbers of Selected Features (5-17) using class_weight technique on credit card dataset*

No. of selected features	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Accuracy	0.999602	0.999590	**0.999637**	0.999614	0.999602	0.999625	0.999625	0.999625	0.999625	0.999590	0.999602	0.999614	0.999579
Precision	0.92	0.93	**0.96**	0.96	0.95	0.97	0.97	0.97	0.96	0.96	0.96	0.96	0.95
Recall	0.82	0.80	**0.81**	0.79	0.79	0.79	0.79	0.79	0.79	0.77	0.79	0.79	0.77
F1-score	0.87	0.86	**0.88**	0.87	0.86	0.87	0.87	0.87	0.87	0.86	0.86	0.87	0.85
ROC-AUC	0.94	0.94	**0.95**	0.96	0.96	0.96	0.96	0.96	0.95	0.95	0.96	0.96	0.95
Macro avg	0.96	0.97	0.98	0.98	0.97	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.98

precision													
Macro avg Recall	0.91	0.90	0.90	0.90	0.90	0.89	0.89	0.89	0.90	0.89	0.89	0.89	0.89
Macro avg F1-score	0.93	0.93	0.94	0.93	0.93	0.93	0.93	0.93	0.94	0.93	0.93	0.93	0.93
Weighted avg precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Weighted avg Recall	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Weighted avg F1-score	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TN	85297	85299	85302	85302	85301	85304	85304	85304	85303	85303	85302	85303	85302
FP	10	8	5	5	6	3	3	3	4	4	5	4	5
FN	24	27	26	28	28	29	29	29	28	31	29	29	31
TP	112	109	110	108	108	107	107	107	108	105	107	107	105

Table 8. *Performance Metrics of the Proposed Model with Numbers of Selected Features (18-30) using class_weight technique on credit card dataset*

No. of selected features	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Accuracy	0.999590	0.999579	0.999579	0.999602	0.999602	0.999625	0.999625	0.999625	0.999614	0.9996374	0.999614	0.9996374	0.999614
Precision	0.96	0.95	0.96	0.96	0.96	0.97	0.97	0.96	0.96	0.97	0.97	0.97	0.96
Recall	0.77	0.77	0.76	0.79	0.78	0.79	0.79	0.79	0.79	0.79	0.78	0.79	0.79
F1-score	0.86	0.85	0.85	0.86	0.86	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
ROC-AUC	0.95	0.96	0.96	0.95	0.97	0.96	0.97	0.96	0.96	0.96	0.95	0.96	0.97
Macro avg precision	0.98	0.98	0.98	0.98	0.98	0.99	0.99	0.98	0.98	0.99	0.99	0.99	0.98
Macro avg Recall	0.89	0.89	0.88	0.89	0.89	0.89	0.89	0.90	0.89	0.90	0.89	0.90	0.89
Macro avg F1-score	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.94	0.93	0.94	0.93	0.94	0.93
Weighted avg precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Weighted avg Recall	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Weighted avg F1-score	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
0TN	85303	85302	85303	85302	85303	85304	85304	85303	85303	85304	85304	85304	85303
FP	4	5	4	5	4	3	3	4	4	3	3	3	4
FN	31	31	32	29	30	29	29	28	29	28	30	28	29
TP	105	105	104	107	106	107	107	108	107	108	106	108	107

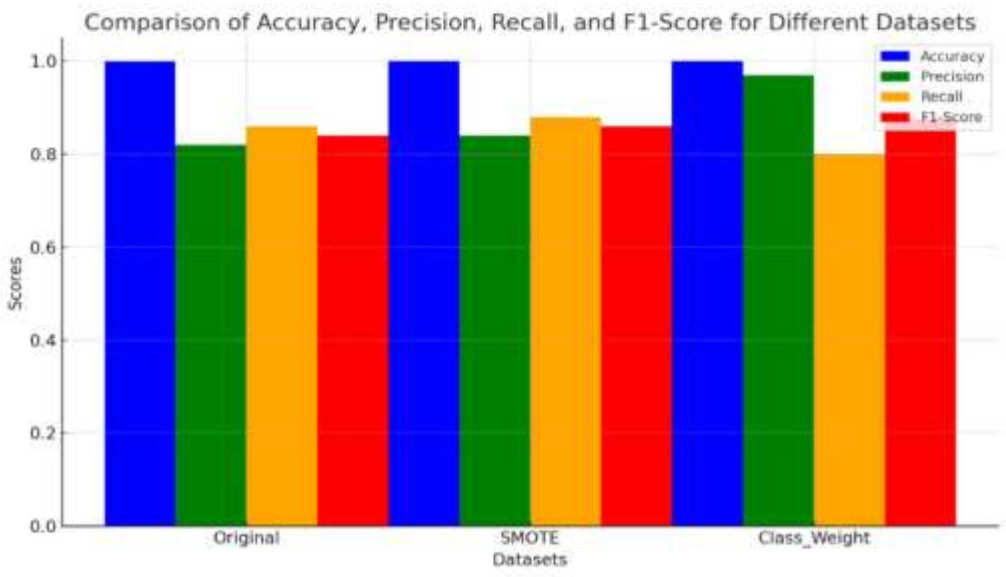

Figure 3. *Visual comparison of accuracy, precision, recall and F1score for original, SMOTE,and class_weight datase*t

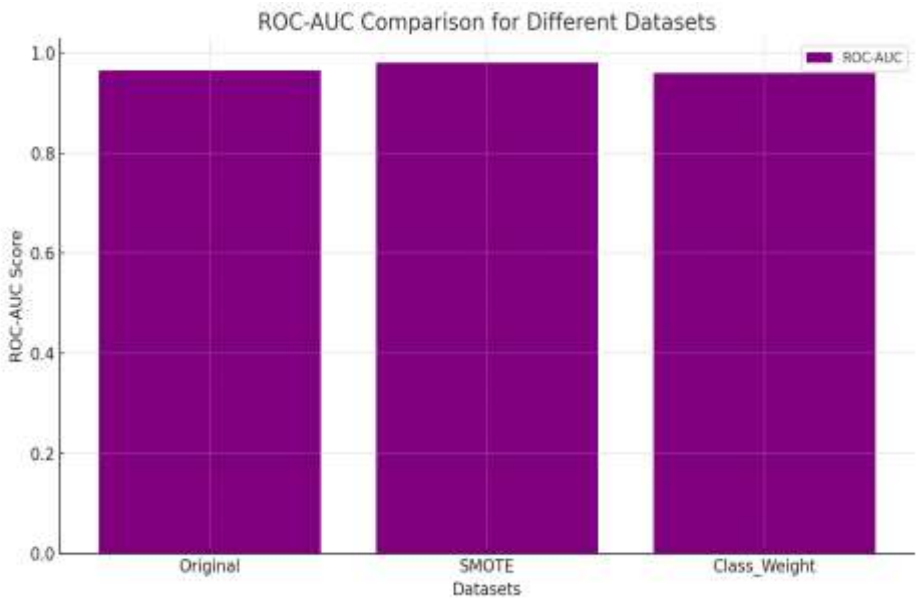

Figure 4. *Visual comparison ROC-AUC for original, SMOTE, and class_weight dataset*

Table 9. *Performance evaluation of Algorithms*

Model	Accuracy	Precision	Recall	F1score	ROC-AUC score
Logistic regression [46]	0.97477	0.0617	0.8730	0.1143	0.9578
LGBM[46]	0.99919	0.7534	0.7990	0.7699	0.9472
XGM[46]	0.99923	0.7862	0.7949	0.7830	0.9517
CatBoost[46]	0.99880	0.6431	0.8096	0.7066	0.9390
Vot_Lg,Xg,Ca[46]	0.99924	0.7720	0.8033	0.7825	0.9501
Vot_Lg, Xg[46]	0.99927	0.7901	0.8012	0.7901	0.9522
Vot_Xg, Ca[46]	0.99923	0.7681	0.8097	0.7823	0.9492
Vot_Lg, Ca[46]	0.99912	0.7260	0.8075	0.7581	0.9459
Deep learning model[46]	0.9994	0.8043	0.8222	0.8132	0.9401
NB [22]	0.9792	–	–	–	–
LR [22]	0.5486	–	–	–	–
k-NN [22]	0.9769	–	–	–	–
DT [25]	0.9708	–	–	–	–
LR [25]	0.9718	–	–	–	–
SVM[26]	0.9512	0.87	–	–	–
LOF[26]	0.99	0.5	–	–	–
iForest[26]	0.99	0.34	–	–	–
DT[26]	0.99	0.00	–	–	–
SVM[27]	0.9750	–	–	–	–
RF[27]	0.9860	–	–	–	–
DT27]	0.9550	–	–	–	–
LR[27]	–	0.9770	–	–	–
DT [28]	–	0.8511	–	–	–
k-NN[28]	–	0.9111	–	–	–
LR[28]	–	0.875	–	–	–
RF[28]	–	0.8977	–	–	–
NB[28]	–	0.652	–	–	–
LOF[29]	0.9960	0.5	–	–	–
IForest[29]	0.9970	0.34	–	–	–
DT[29]	0.9980	0.00	–	–	–
XGBoost [30]	0.9996	0.9773	0.8269	0.8958	–
XGBoost + Random Oversampling[30]	0.9996	0.9663	0.8269	0.8912	–
XGBoost + SMOTE[30]	0.9995	0.8969	0.8365	0.8657	–
XGBoost + SMOTETomek[30]	0.9995	0.8700	0.8365	0.8529	–
XGBoost + SMOTEENN[30]	0.9995	0.8627	0.8462	0.8544	–
XGBoost + ADASYN[30]	0.9994	0.8529	0.8365	0.8447	–
DT[31]	0.96	–	–	–	–
LR[31]	0.95	–	–	–	–
k-NN[31]	0.97	–	–	–	–
RF[31]	0.97	–	–	–	–
XGBoost[31]	0.96	–	–	–	–
NB[31]	0.94	–	–	–	–
SVM[31]	0.95	–	–	–	–
RF_SGO on original dataset	0.999672	0.96	0.83	0.89	0.96
RF_SGO after SMOTE	0.999544	0.84	0.88	0.86	0.98
RF_SGO with class_weight	0.999637	0.96	0.81	0.88	0.95

V28, along with two non-transformed features, “Time” and “Amount.”

- The “Time” feature records the time in seconds between a transaction and the first transaction in the dataset.
- The “Amount” feature represents the monetary value of the transaction.
- The target variable, “Class” indicates the outcome of the transaction, where 1 represents a fraudulent transaction and 0 represents a legitimate one.

A summary of the dataset's features and variables is provided in Table 1.

Simulation results: Performance of RF_SGO model in predicting frauds on creditcard dataset

To evaluate the performance of the proposed RF_SGO model, we utilized three variations of the original creditcard dataset: (1) the original creditcard dataset(unbalanced) (2) the dataset balanced using the SMOTE technique, and (3) the dataset with class_weight adjustments (unbalanced). Detailed descriptions of these datasets are provided in Table 2.

Performance of the RF_SGO model on original credit card dataset

The performance metrics of the RF_SGO model were computed for varying numbers of selected features. Results for feature sets ranging from 5 to 17 are presented in Table 3, while results for feature sets ranging from 18 to 30 are shown in Table 4, based on the original credit card dataset.

Discussion

Tables 3 and 4 present the performance metrics of the RF_SGO model on the original credit card dataset, evaluated for varying numbers of selected features (5–30). Below, we analyze the key trends and insights from these results.

Accuracy

Across all feature subsets, the RF_SGO model achieves exceptionally high accuracy, ranging between 0.999579 and 0.999672. This consistency indicates that the model performs well in correctly classifying the majority class (legitimate transactions), which is expected given the dataset's imbalance. Slight variations in accuracy are observed as the number of features changes, with the highest accuracy (0.999672) achieved with 25 selected features.

Precision

Precision values range from 0.91 to 0.96. Precision improves as the number of features increases, reflecting the model’s ability to minimize false positives (legitimate transactions misclassified as fraudulent). Notably, the highest precision values (**0.96**) are achieved at multiple feature counts (e.g., 10, 19, 23, 28).

Recall

Recall varies between 0.79 and 0.83, showing more fluctuation compared to other metrics. Recall is crucial for fraud detection, as it indicates the proportion of actual fraudulent transactions correctly identified. Although recall does not demonstrate a consistent increasing trend, the highest value (0.83) is observed with 25 selected features, suggesting better sensitivity to the minority class.

F1-Score

The F1-score, a balance between precision and recall, ranges from 0.85 to 0.89. Similar to precision, the F1-score improves with a larger feature subset, peaking at 0.89 for 25 features. This suggests that the RF_SGO model achieves its best trade-off between minimizing false positives and false negatives with this feature count.

ROC-AUC

The ROC-AUC values are consistently high (between 0.94 and 0.97), reflecting the model's strong capability to distinguish between legitimate and fraudulent transactions. The highest value (0.97) is observed with 28 features, indicating optimal discriminatory power at this feature count.

Macro Average Metrics: Precision, Recall, and F1-score (Macro Average):

Macro average precision consistently exceeds **0.97**, demonstrating the model's strong performance across both classes. Recall, however, remains stable around 0.90–0.92, reflecting challenges in boosting minority class detection. Macro average F1-scores show steady improvements with feature count, reaching 0.94 for larger subsets. These macro averages validate that performance improvements are achieved without compromising the minority class entirely, despite the dataset imbalance.

Weighted Average Metrics

Weighted average precision, recall, and F1-score remain constant at **1.00** across all feature subsets, highlighting that the model performs nearly perfectly on the majority class. However, these metrics should be interpreted carefully, as they are heavily influenced by the majority class's overwhelming presence.

Confusion Matrix (TN, FP, FN, TP)

True Negatives (TN): The TN values are consistently high across all feature subsets, reflecting the model’s strength in identifying legitimate transactions correctly.

False Positives (FP): FP values decrease as the number of features increases, with the
lowest FP count (4) observed at 17 and 29 features.

False Negatives (FN): FN values fluctuate between 23 and 28, indicating room for improvement in correctly identifying fraudulent transactions.

True Positives (TP): TP values remain relatively stable, peaking at 113 with 25 features, which aligns with the highest recall observed.

Insights and Observations

Optimal Feature Selection:

The RF_SGO model achieves its best overall performance in terms of accuracy, precision, recall, F1-score, and ROC-AUC with feature counts between 23–28. A feature count of 25 appears to be particularly effective, yielding the highest recall (0.83) and strong results across all metrics.

Class Imbalance Challenges:

The model demonstrates high precision but relatively lower recall, indicating that while it minimizes false positives effectively, there is room to improve its sensitivity to fraudulent transactions (reducing FN).

Trade-offs Between Precision and Recall:

The increase in precision with higher feature counts comes at a slight cost to recall in some cases. This trade-off needs to be balanced based on the application context-whether minimizing false alarms or detecting all fraud cases is prioritized.

Real-World Applicability:

Despite the dataset's class imbalance, the RF_SGO model maintains strong performance, suggesting its potential for real-world deployment. However, emphasis should be placed on improving recall to ensure the model effectively detects fraudulent transactions in critical scenarios.

Tables 3 and 4 demonstrate the RF_SGO model's robustness across varying feature subsets. While the model performs exceptionally well overall, further optimization- such as feature engineering or adjustments to the training process-could focus on enhancing recall for more reliable fraud detection in real-world applications.

Performance of the RF_SGO model on SMOTE-treated credit card dataset

The performance metrics of the RF_SGO model were computed for varying numbers of selected features. Results for feature sets ranging from 5 to 17 are presented in Table 5, while results for feature sets ranging from 18 to 30 are shown in Table 6, based on the credit card dataset after SMOTE oversampling.

Tables 5 and 6 provide a comprehensive analysis of the RF_SGO model's performance using SMOTE oversampling, varying the number of selected features from 5 to 30. The discussion below highlights key trends and insights drawn from the metrics.

Accuracy

Accuracy improves progressively from 0.9976 (5 features) to a peak of 0.999544 (17 features). This trend demonstrates that adding features enhances the model's ability to generalize while leveraging the oversampled dataset. Beyond 17 features, accuracy remains stable, ranging between 0.999497 and 0.999544. This suggests that adding more features beyond 17 provides limited additional value, as the model likely captures most relevant information by this point.

Precision

In Table 5, precision starts low at 0.39 (5 features) and steadily increases, reaching 0.84 at 17 features. In Table 6, precision stabilizes around 0.82–0.84 for feature counts between 18 and 30. Precision growth reflects the model's ability to reduce false positives with more features. However, diminishing returns are evident as the feature count exceeds 17.

Recall

Recall values remain steady across Tables 5 and 6, staying within the range of 0.87–0.89. The balanced dataset ensures the model consistently identifies fraudulent transactions regardless of the number of features. While recall remains high, slight variations occur as precision improves, indicating an inherent trade-off between these metrics as more features are introduced.

F1-Score

F1-score increases from 0.54 (5 features) to a high of 0.86 (17 features), reflecting improved balance between precision and recall. Beyond 17 features, F1-scores plateau around 0.85, indicating that the addition of more features does not further enhance the balance of detection capabilities.

ROC-AUC

ROC-AUC remains consistently high (0.97–0.98) across both tables, demonstrating the model's ability to effectively distinguish between fraudulent and non-fraudulent transactions, even with varying feature sets.

Macro and Weighted Averages

Macro-averaged precision and F1-scores improve up to 17 features, stabilizing thereafter. This highlights the model's robustness in treating both classes equally. Weighted average precision, recall, and F1-scores remain perfect (1.00) across all feature counts. This reflects the significant dominance of non-fraudulent transactions and the model's strong performance on the majority class.

Confusion Matrix Analysis

True Negatives (TN) and False Positives (FP): TN values increase slightly as more features are added, reducing FP rates from 189 (5 features) to 23 (17 features in Table 5) and stabilizing in Table 6. This indicates improved classification of non-fraudulent transactions.

True Positives (TP) and False Negatives (FN): TP values slightly fluctuate but remain steady at ~120, while FN values stay low, varying between 15 and 18. This reflects consistent sensitivity to fraudulent cases across different feature sets.

Key Insights

Optimal Feature Count:

Performance metrics (accuracy, precision, F1-score) improve significantly with features up to 17, with diminishing returns beyond this point. Therefore, 15–17 features represent an optimal balance between performance and complexity.

Impact of SMOTE:

SMOTE effectively balances the dataset, enabling the model to maintain high recall and F1-scores. This is crucial for detecting rare fraudulent transactions.

Practical Implications:

In real-world scenarios, recall and F1-score are critical for fraud detection to minimize undetected fraudulent transactions. A feature set of 15–17 offers the best trade-off between performance and computational efficiency.

Scalability:

The model demonstrates consistent performance across a wide range of feature counts, showcasing its scalability and adaptability for datasets with different feature dimensions.

The RF_SGO model, when applied to a SMOTE-treated credit card dataset, achieves high precision, recall, and F1-scores with 15–17 selected features. This ensures robust and balanced fraud detection capabilities, making it well-suited for practical applications where minimizing false negatives is critical.

Performance of the RF_SGO model using the class_weight technique on credit card dataset

The performance metrics of the RF_SGO model were computed for varying numbers of selected features. Results for feature sets ranging from 5 to 17 are presented in Table 7, while results for feature sets ranging from 18 to 30 are shown in Table 8, based on the credit card dataset using class_weight technique.

Tables 7 and 8 present the performance of the RF_SGO model applied to the credit card dataset with feature subsets ranging from 5 to 30, optimized using the class_weight technique. Key insights are discussed below:

Accuracy

The model achieves consistently high accuracy, exceeding 99.95% for all feature subsets. Accuracy peaks at 0.999637 for 7 features, showing that smaller subsets can yield excellent results. Beyond 10 features, accuracy remains relatively stable, fluctuating slightly around 0.999590–0.999625. Accuracy remains consistent with minor improvements at 27 and 29 features (0.999637). These results demonstrate the robustness of the model across feature subset sizes.

Precision

Precision remains high across all subsets, indicating the model's ability to minimize false positives. Precision improves from 0.92 (5 features) to a maximum of 0.97 (10–12 features), maintaining this peak for most subsets. Precision remains consistently high between 0.95 and 0.97, showing marginal gains with larger feature subsets.

Recall

Recall, which measures the model's ability to identify fraudulent transactions, shows variability: Recall is highest at 0.82 (5 features) but declines slightly to 0.77 (17 features) as the feature subset increases. Recall stabilizes between 0.76 and 0.79, demonstrating that the model maintains a good sensitivity to fraudulent transactions despite adding features.

F1-Score

The F1-score, balancing precision and recall, is consistently strong: Peaks at 0.88 (7 features) and remains stable around 0.87 for most feature subsets. F1-score fluctuates between 0.85 and 0.87, reflecting robust overall performance across larger feature subsets.

ROC-AUC

The ROC-AUC score, representing the model's discriminatory power, remains excellent: Increases slightly from 0.94 (5 features) to 0.96 (10–12 features), confirming strong model performance with smaller subsets. Scores stabilize between 0.95 and 0.97, indicating reliable separation between classes even with larger feature subsets.

Confusion Matrix Observations

True Negatives (TN): The model consistently identifies legitimate transactions, with TN values exceeding 85,300 across all subsets.

False Positives (FP): FP values remain low, particularly in optimal subsets: Minimum FP values occur with 10–12 features (only 3 false positives). FP stabilizes around 3–5, reinforcing the model's ability to minimize false alarms.

True Positives (TP): TP values range between 104 and 112, reflecting consistent identification of fraudulent transactions.

False Negatives (FN): FN values slightly increase with feature count: FN ranges from 24 (5 features) to 31 (17 features). FN values stabilize between 28 and 32, indicating slight limitations in capturing all fraudulent instances.

Macro-Averaged Metrics

Macro-Averaged Precision and Recall: Precision increases with feature count, peaking at 0.99 for 10–12 features, while recall stabilizes around 0.89–0.91. Precision remains between 0.98 and 0.99, with stable recall values (0.89–0.90), showcasing balanced performance across classes.

Macro-Averaged F1-Score: Remains consistent at 0.93–0.94, confirming the model's robustness across subsets.

Weighted Metrics

Weighted Precision, Recall, and F1-Scores are perfect (1.0) for all feature subsets in both tables, emphasizing the model's ability to handle imbalanced datasets effectively.

Smaller subsets (7–12 features) achieve the best trade-off between precision, recall, and F1-score, with fewer false positives and slightly higher recall. Larger subsets (18–30 features) maintain stable performance but show diminishing returns in terms of recall and F1-score. The RF_SGO model, combined with the class_weight technique, demonstrates exceptional performance across all metrics, handling the imbalanced nature of the dataset effectively while ensuring minimal computational overhead for smaller subsets. Future research can explore fine-tuning feature subsets further to optimize recall while maintaining precision.

Comparison on evaluating the original dataset, SMOTE Oversampling dataset, and Class_Weight dataset to determine the best-performing dataset for RF_SGO

The visual summary (charts) and additional insights to give a complete view of the RF_SGO performance across the different datasets is explained below:

Accuracy, Precision, Recall, and F1-Score Comparison:

The comparison chart shows how these metrics change for each dataset across the feature ranges. Class Weight has the highest Precision and F1-Score, with values around 0.97 and 0.87, respectively, which indicate that the model using class weight performs the best in terms of correctly identifying positive instances (fraud cases). SMOTE has slightly better Recall than Class Weight (around 0.88), suggesting that SMOTE provides a better balance in terms of detecting both fraud and non-fraud instances. The Original Dataset generally performs well but trails behind the others in Precision and Recall.

ROC-AUC Comparison:

The SMOTE and Class Weight datasets both achieve high ROC-AUC scores of around 0.98 and 0.96, respectively. The Original Dataset has a slightly lower ROC-AUC (~0.96). A higher ROC-AUC score reflects better model performance, so SMOTE seems to provide the best overall performance, followed closely by Class Weight.**Additional Insights:**

Based on both accuracy and precision/recall balance, SMOTE appears to be the best dataset for the RF_SGO model. It offers the highest ROC-AUC and better F1-Score, making it the most suitable dataset for distinguishing between fraud and non-fraud cases. While Class Weight provides excellent Precision and F1-Score, it tends to have slightly lower Recall than SMOTE, indicating it might be slightly more conservative in detecting fraud instances. However, it could be useful in scenarios where false positives need to be minimized (e.g., avoiding too many false alarms). The Original Dataset does perform well, but it lags behind SMOTE and Class Weight across most metrics. It's generally better suited for baseline comparisons or when data balancing techniques like SMOTE or class weighting aren't available.

The SMOTE dataset offers the best overall performance, balancing both detection accuracy and model sensitivity, making it the preferred choice for fraud detection in this case. However, depending on the application, Class Weight might be a better choice if minimizing false positives is a priority.

Simulation results: Performance comparisons in predicting frauds on creditcard data by various algorithms

Hyperparameters have a significant effect on the performance of machine learning models. We refer to optimization as the process of finding the best set of hyperparameters that configure a machine learning algorithm during its training. Recently, it was shown that the evolutionary optimization methods is capable of finding the optimised values in a much smaller number of training courses compared with traditional optimization methods [44-45]. In this paper, we use the SGO optimization algorithm to tune the hyperparameters of RF that leads to performance improvement. For comparison purpose we have utilized various results of algorithms imported from various research paper which are listed as Logistic regression [46], LGBM[46], XGM[46], CatBoost[46], Vot_Lg,Xg,Ca[46], Vot_Lg, Xg[46], Vot_Xg, Ca[46], Vot_Lg, Ca[46], Deep learning model[46], NB [22], LR [22], *k*-NN [22], DT [25], LR [25], SVM[26], LOF[26], iForest[26], DT[26], SVM[27], RF[27], DT[27], LR[27], DT[28], k-NN[28], LR[28], RF[28], NB[28], LOF[29], iForest[29], DT[29], XGBoost [30], XGBoost + Random Oversampling[30], XGBoost + SMOTE[30], XGBoost + SMOTETomek[30], XGBoost + SMOTEENN[30], XGBoost + ADASYN[30], DT[31], LR[31], *k*-NN[31], RF[31], XGBoost[31], NB[31], SVM[31].***Discussion***

The table 9 presents a comparative performance evaluation of various machine learning models for creditcard fraud detection, using key metrics such as Accuracy, Precision, Recall, F1-Score,
and ROC-AUC score. This comparison includes traditional machine learning algorithms, advanced ensemble methods, deep learning models, and models optimized using the SGO algorithm.

LR [46] achieves an Accuracy of 0.97477 but performs poorly in Precision (0.0617) despite a high

Recall of 0.8730. This imbalance suggests that LR tends to overpredict the positive class, leading to a high false-positive rate. LR models from other studies [22], [27], [28], [31] show variable results, but none outperform ensemble methods or RF_SGO. DT models from [29], [25], [26], [27], [28], and [31] exhibit inconsistent performance, with Accuracy ranging from 0.9550 to 0.9980. However, DTs typically have lower precision, indicating suboptimal performance for imbalanced datasets. The k-NN models [22], [28], [31] perform moderately well with Accuracy around 0.97, but detailed precision and recall values are unavailable, making it difficult to assess their suitability for fraud detection. SVM models [26], [31] achieve Accuracy up to 0.9750 and moderate precision (0.87). However, they are not consistently better than ensemble methods or RF_SGO, particularly in recall. NB models [22], [28], [31] achieve lower accuracy (0.94–0.9792), indicating their limited capability for handling this imbalanced dataset.

RF models from [27], [28], and [31] show Accuracy up to 0.9860, indicating good generalization. However, RF_SGO outperforms vanilla RF in both precision and recall due to its hyperparameter optimization. XGBoost models [30] perform well, with Accuracy reaching 0.9996. Variants like SMOTE and SMOTEENN enhance recall and F1-scores but do not surpass RF_SGO's overall balance of metrics. For instance, XGBoost with random oversampling achieves Precision of 0.9663 and F1-Score of 0.8912, slightly below RF_SGO (SMOTE). CatBoost [46] achieves Accuracy of 0.99880, with strong recall (0.8096) but lower precision (0.6431). LGBM [46] performs slightly better with higher precision (0.7534) but still lags behind RF_SGO. Ensemble combinations like Vot_Lg, Xg [46] and Vot_Xg, Ca [46] achieve Accuracy around 0.9992–0.9993, with precision and recall values similar to XGBoost but lower than RF_SGO.

RF_SGO achieves the highest Accuracy (0.999672) and Precision (0.96), making it ideal for scenarios prioritizing precision and minimizing false alarms. However, the Recall (0.83) is slightly lower compared to SMOTE-enhanced variants. The SMOTE-enhanced RF_SGO model achieves the highest ROC-AUC (0.98) with improved Recall (0.88), balancing the trade-off between precision and recall. This variant is ideal for scenarios where detecting fraudulent transactions is critical. The class-weighted RF_SGO model has an Accuracy of 0.999637 and strong Precision (0.96), but its Recall (0.81) is the lowest among RF_SGO variants, indicating that it is less effective at identifying minority class instances.

The deep learning model [46] achieves competitive metrics, with Accuracy of 0.9994, Precision of 0.8043, and F1-Score of 0.8132. However, its lower ROC-AUC (0.9401) compared to RF_SGO indicates suboptimal performance in distinguishing classes.

The RF_SGO with SMOTE variant is the most balanced model, achieving the highest ROC-AUC (0.98) and a robust trade-off between precision and recall. The RF_SGO on the original dataset is best suited for applications prioritizing precision (e.g., minimizing false positives). Logistic regression, decision trees, and naïve Bayes are less effective due to the imbalanced nature of the dataset, often favoring majority class predictions. Models like XGBoost and its variants, CatBoost, and ensemble methods are competitive but do not surpass RF_SGO's performance, especially in recall and AUC. When compared to traditional models like Logistic Regression (LR), Naive Bayes (NB), and Decision Trees (DT), RF_SGO outperforms them on all major metrics, especially in terms of Precision, Recall, and ROC-AUC. The SMOTE-enhanced RF_SGO achieves the highest ROC-AUC (0.98) and Recall (0.88), providing the best overall balance between detecting fraud and minimizing false positives. The visual comparison is illustrated by the figures 3-9.

RF_SGO after SMOTE is the most balanced model across all metrics, achieving a high ROC-AUC of 0.98, excellent recall (88%), and a reasonable precision (84%). This version of the model is ideal for scenarios where identifying fraudulent transactions is the highest priority. RF_SGO on the original dataset is best for applications where minimizing false positives is crucial, as it achieves a very high precision (96%) but slightly sacrifices recall (83%). Class-weighted RF_SGO strikes a balance between the two extremes, providing excellent precision (96%) while still maintaining decent recall (81%). This configuration is well-suited for situations where both precision and recall are important but precision is prioritized. XGBoost and LGBM perform well, but their performance is outpaced by RF_SGO variants in terms of the precision-recall trade-off and ROC-AUC. Ensemble methods (e.g., Vot_Lg, Xg, Ca) and deep learning models perform competitively but generally fall short of RF_SGO’s ability to effectively balance precision and recall for fraud detection tasks. This performance evaluation shows that RF_SGO, particularly when combined with SMOTE, provides a highly effective approach for credit card fraud detection, outshining many traditional and advanced models in various key metrics.

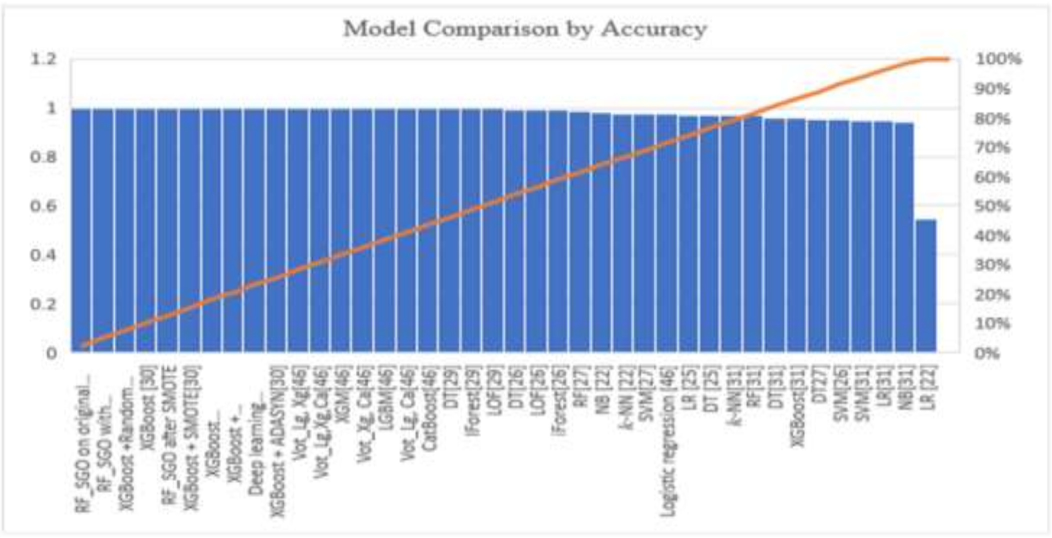

Figure 5. *Visual comparison of Accuracy by different algorithms*

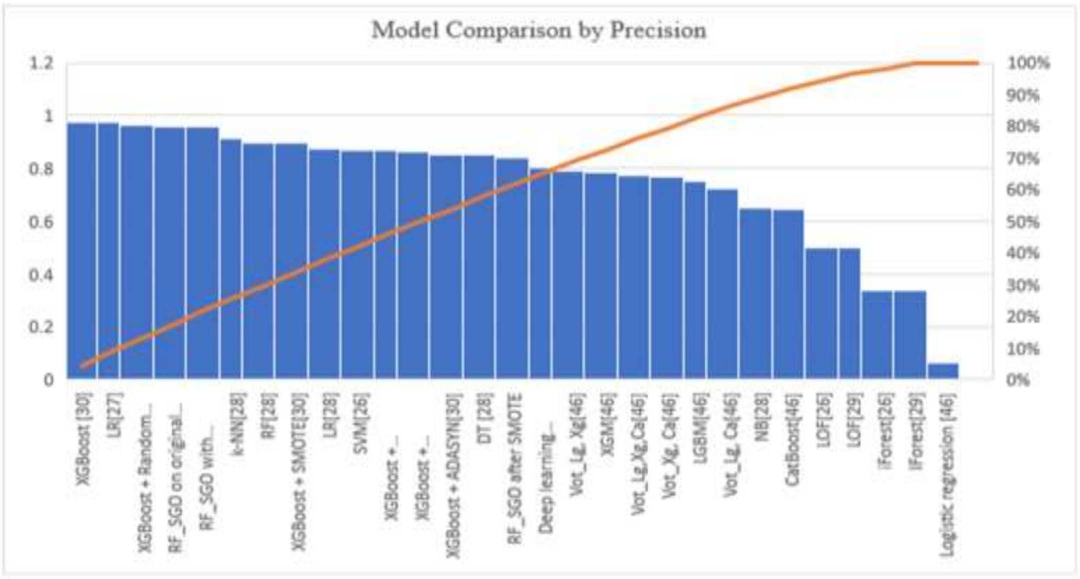

Figure 6. *Visual comparison of Precision by different algorithms*

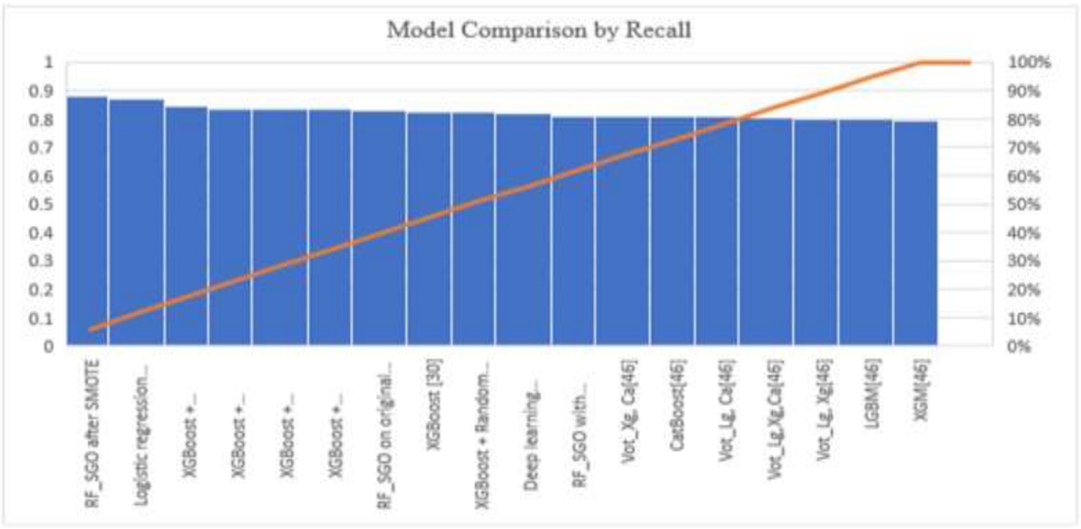

Figure 7. *Visual comparison of Recall by different algorithms*

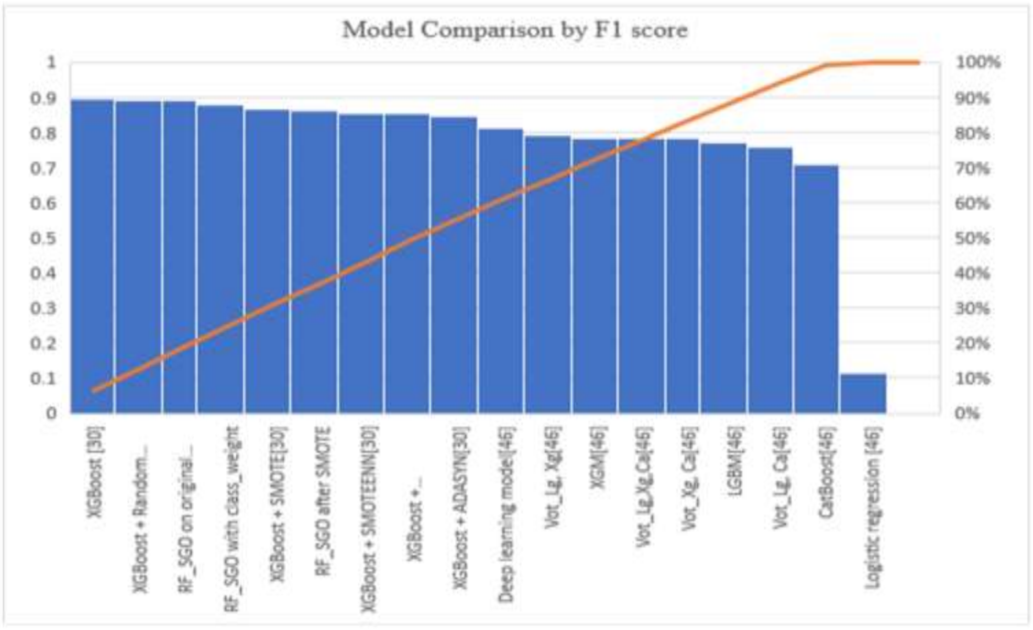

Figure 8. *Visual comparison of F1 score by different algorithms*

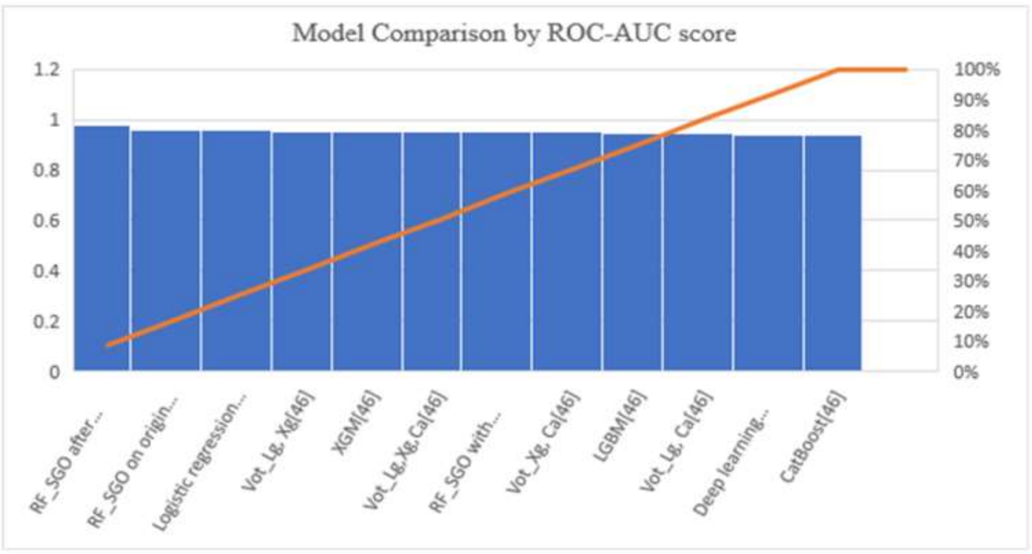

Figure 9. *Visual comparison of ROC-AUC score by different algorithms*

7. Conclusion and Future work

This study demonstrates the significant potential of the RF_SGO framework for credit card fraud detection, particularly in handling highly imbalanced datasets. Among the tested approaches, the RF_SGO model combined with SMOTE emerges as the most balanced, achieving an impressive ROC-AUC of 0.98, high recall (88%), and reasonable precision (84%). This configuration is particularly suited for scenarios where identifying fraudulent transactions is the top priority. In contrast, the class-weighted RF_SGO model maintains excellent precision (96%) with decent recall (81%), making it ideal for applications requiring a balance between sensitivity and specificity. The RF_SGO model applied to the original dataset, with its very high precision (96%) and slightly lower recall (83%), is optimal for minimizing false positives. Additionally, the integration of feature importance analysis enhances the interpretability of the Random Forest model by identifying key predictors of fraudulent behavior, providing actionable insights for financial institutions. The comparative evaluation further highlights RF_SGO's superior performance over traditional machine learning models (e.g., Logistic Regression, Decision Trees, and SVM), advanced algorithms (e.g., XGBoost, CatBoost), and deep learning techniques in terms of precision-recall trade-offs and ROC-AUC metrics.The findings emphasize the importance of hyperparameter optimization, feature importance analysis, and effective data balancing techniques in building robust and scalable fraud detection systems. These contributions provide financial institutions with a practical and efficient framework to mitigate credit card fraud, ensuring secure transaction environments. While this study establishes RF_SGO as a highly effective solution, several areas for future exploration remain. Implementing RF_SGO in real-time fraud detection systems to evaluate its performance in dynamic and evolving environments. Enhancing the framework's interpretability using advanced explainable AI techniques to meet regulatory requirements and build stakeholder trust. Extending the evaluation to diverse datasets across industries to validate the generalizability of the proposed framework. Combining RF_SGO with other advanced techniques, such as ensemble methods and deep learning, to further improve detection accuracy.By addressing these directions, the RF_SGO framework can evolve into a more adaptable, reliable, and comprehensive solution for combating fraudulent activities in the financial sector.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Nanduri, J., Liu, Y.-W., Yang, K., & Jia, Y. (2020). Ecommerce fraud detection through fraud islands and multi-layer machine learning model. *In Future of Information and Communication Conference* (pp. 556–570). Springer.

[2] Matloob, I., Khan, S. A., Rukaiya, R., Khattak, M. A. K., & Munir, A. (2022). A sequence mining-based novel architecture for detecting fraudulent transactions in healthcare systems. *IEEE Access, 10,* 48447–48463. https://doi.org/10.1109/ACCESS.2022.3171418

[3] Sulaiman, B. R., Schetinin, V., & Sant, P. (2022). Review of machine learning approach on credit card fraud detection. *Human-Centric Intelligent Systems, 2,* 55–68. https://doi.org/10.1007/s44230-022-00004-0

[4] Dornadula, V. N., & Geetha, S. (2019). Credit card fraud detection using machine learning algorithms. *Procedia Computer Science, 165,* 631–641. https://doi.org/10.1016/j.procs.2020.01.057

[5] Sekar, M. (2022). Fraud and anomaly detection. In *Machine Learning for Auditors* (pp. 321–340). Apress. https://doi.org/10.1007/978-1-4842-8051-5_21

[6] Kaggle. (n.d.). The credit card fraud. *Kaggle*. Retrieved from https://www.kaggle.com/mlg-ulb/creditcardfraud

[7] Satapathy, S., & Naik, A. (2016). Social group optimization (SGO): A new population evolutionary optimization technique. *Complex & Intelligent Systems, 2*(3), 173–203.

[8] Naik, A., et al. (2018). Social group optimization for global optimization of multimodal functions and data clustering problems. *Neural Computing & Applications, 30*(1), 271–287. https://doi.org/10.1007/s00521-016-2614-6

[9] Naik, A., & Chokkalingam, P. K. (2022). Binary social group optimization algorithm for solving 0-1 knapsack problem. *Decision Science Letters, 11*(1), 55–72.

[10] Monisha, R., et al. (2019). Social Group Optimization and Shannon's Function-Based RGB Image Multi-level Thresholding. In *Smart Intelligent Computing and Applications* (pp. 123–132). Springer, Singapore.

[11] Reddy, A., & Narayana, K. V. L. (2022). Investigation of a multi-strategy ensemble social group optimization algorithm for the optimization of energy management in electric vehicles. *IEEE Access, 10,* 12084–12124.

[12] Manic, K. S., Al Shibli, N., & Al Sulaimi, R. (2018). SGO and Tsallis entropy-assisted segmentation of abnormal regions from brain MRI. *Journal of Engineering Science and Technology, 13,* 52–62.

[13] Parwekar, P. (2018). SGO: A new approach for energy efficient clustering in WSN. *International Journal of Natural Computing Research, 7*(3), 54–72.

[14] Pant, M., et al. (2008). Improved particle swarm optimization with low-discrepancy sequences. In *2008 IEEE Congress on Evolutionary Computation* (Vols 1–8, pp. 3011–3018).

[15] Naik, A., Jena, J. J., & Satapathy, S. C. (2021). Non-dominated sorting social group optimization algorithm for multi-objective optimization. *Journal of Scientific & Industrial Research*.

[16] Naik, A. (2023). Chaotic social group optimization for structural engineering design problems. *Journal of Bionic Engineering, 20,* 1852–1877. https://doi.org/10.1007/s42235-023-00340-2

[17] Naik, A. (2024). Marine predators social group optimization: A hybrid approach. *Evolutionary*

Intelligence, 17, 2355–2386. https://doi.org/10.1007/s12065-023-00891-7

[18] Naik, A. (2024). Multi-objective social group optimization for machining process. *Evolutionary Intelligence, 17*, 1655–1676. https://doi.org/10.1007/s12065-023-00856-w

[19] Campus, K. (2018). Credit card fraud detection using machine learning models and collating machine learning models. *International Journal of Pure and Applied Mathematics, 118*(20), 825–838.

[20] Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M., & Anderla, A. (2019). Credit card fraud detection-machine learning methods. In *18th International Symposium INFOTEH-JAHORINA* (pp. 1–5).

[21] Khatri, S., Arora, A., & Agrawal, A. P. (2020). Supervised machine learning algorithms for credit card fraud detection: A comparison. In *10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 680–683).

[22] Awoyemi, J. O., Adetunmbi, A., & Oluwadare, S. (2018). Effect of feature ranking on the detection of credit card fraud: Comparative evaluation of four techniques. *I-Manage Journal of Pattern Recognition, 5*(3), 10.

[23] Dornadula, V. N., & Geetha, S. (2019). Credit card fraud detection using machine learning algorithms. *Procedia Computer Science, 165*, 631–641.

[24] Seera, M., Lim, C. P., Kumar, A., Dhamotharan, L., & Tan, K. H. (2021). An intelligent payment card fraud detection system. *Annals of Operations Research*, 1–23.

[25] Khalilia, M., Chakraborty, S., & Popescu, M. (2011). Predicting disease risks from highly imbalanced data using random forest. *BMC Medical Informatics and Decision Making, 11*, 1–13.

[26] Rtayli, N., & Enneya, N. (2020). Selection features and support vector machine for credit card risk identification. *Procedia Manufacturing, 46*, 941–948.

[27] Ileberi, E., Sun, Y., & Wang, Z. (2022). A machine learning based credit card fraud detection using the GA algorithm for feature selection. *Journal of Big Data, 9*(1), 1–17.

[28] Khan, M. Z., Shaikh, S. A., Shaikh, M. A., Khatri, K. K., Rauf, M. A., Kalhoro, A., Adnan, M. (2022). The performance analysis of machine learning algorithms for credit card fraud detection. *International Journal of Online Engineering, 19*(03), 83.

[29] Agarwal, A., & Ratha, N. K. (2021). Black-box adversarial entry in finance through credit card fraud detection. In *CIKM Workshops*.

[30] Noviandy, T. R., Idroes, G. M., Maulana, A., Hardi, I., Ringga, E. S., & Idroes, R. (2023). Credit card fraud detection for contemporary financial management using XGBoost driven machine learning and data augmentation techniques. *Indatu Journal of Management and Accounting, 1*(1), 29–35.

[31] Sinap, V. (2024). Comparative analysis of machine learning techniques for credit card fraud detection: Dealing with imbalanced datasets. *Turkish Journal of Engineering, 8*(2), 196–208.

[32] Naik, A., Satapathy, S. C., & Abraham, A. (2020). Modified Social Group Optimization—a meta-heuristic algorithm to solve short-term hydrothermal scheduling. *Applied Soft Computing, 95*, 106513. https://doi.org/10.1016/j.asoc.2020.106513

[33] Chawla, N., Bowyer, K., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *ArXiv*, abs/1106.1813.

[34] He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

[35] Archer, K. J., & Kimes, R. V. (2008). Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis, 52*(4), 2249–2260. https://doi.org/10.1016/j.csda.2007.08.015

[36] Powers, D. M. (2020). Evaluation: From precision, recall, and F-measure to ROC, informedness, markedness, and correlation. *arXiv preprint arXiv:2010.16061*.

[37] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

[38] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

[39] Scikit-learn User Guide. (n.d.). Retrieved from https://scikit-learn.org/stable/user_guide.html

[40] Pajankar, A., & Joshi, A. (2022). Getting started with NumPy. In *Hands-on Machine Learning with Python* (pp. 23–30). Apress. https://doi.org/10.1007/978-1-4842-7921-2_2

[41] Matplotlib Overview. (n.d.). Retrieved from https://matplotlib.org/stable/contents.html

[42] Pajankar, A., & Joshi, A. (2022). Introduction to pandas. In *Hands-on Machine Learning with Python* (pp. 45–61). Apress. https://doi.org/10.1007/978-1-4842-7921-2_4

[43] Waskom, M. (n.d.). Seaborn User Guide and Tutorial. Retrieved from https://seaborn.pydata.org/tutorial.html

[44] Nistor, S. C., & Czibula, G. (2022). IntelliSwAS: Optimizing deep neural network architectures using a particle swarm-based approach. *Expert Systems with Applications, 187*, 115945.

[45] Ghosh, A., Jana, N. D., Mallik, S., & Zhao, Z. (2022). Designing optimal convolutional neural network architecture using differential evolution algorithm. *Patterns, 3*(9), 100567.

[46] Hashemi, S. K., Mirtaheri, S. L., & Greco, S. (2023). Fraud detection in banking data by machine learning techniques. *IEEE Access, 11*, 3034–3043. https://doi.org/10.1109/ACCESS.2023.3275174