

Leak-Free Credit Card Fraud Detection Through Fold-Isolated Data Mining and Threshold-Calibrated Deep Learning

Rukundo Evariste

Faculty of Computing and
Information Sciences

University of Lay Adventists
of Kigali (UNILAK)

Kigali, Rwanda

Enan Muhire Nyesheja

Faculty of Computing and
Information Sciences

University of Lay Adventists
of Kigali (UNILAK)

Kigali, Rwanda

Mungwarakarama Irene

Faculty of Computing and
Information Sciences

University of Lay Adventists
of Kigali (UNILAK)

Kigali, Rwanda

Ntivuguruzwa Jean De La
Croix

Faculty of Computing and
Information Sciences,
University of Lay Adventists
of Kigali (UNILAK), Kigali,
Rwanda

Department of Business
Information Technology,
University of Rwanda,
Kigali, Rwanda

jc.ntivuguruzwa@ur.ac.rw

Corresponding Author: jc.ntivuguruzwa@ur.ac.rw

Abstract—Credit card fraud detection remains challenging because fraudulent transactions represent less than 0.2% of all transactions, creating severe class imbalance. Existing studies often apply feature selection and SMOTE without clearly specifying whether these operations are performed independently within cross-validation folds, which creates a risk of data leakage and inflated performance estimates. This study proposes a fold-quarantined iterative Knowledge Discovery in Databases (KDD) pipeline in which feature selection, SMOTE, feature scaling, model training, and decision-threshold calibration are performed exclusively on the training data of each fold. The pipeline is evaluated with stratified five-fold cross-validation on the Kaggle Credit Card Fraud Detection dataset (284,807 transactions, 492 of them fraudulent, 0.172%) using a Multilayer Perceptron (MLP) trained with focal loss, Logistic Regression, Decision Tree, and Random Forest. Random Forest achieved the best overall performance ($F1 = 0.843$), followed by the focal-loss MLP ($F1 = 0.815$), whereas the Decision Tree and Logistic Regression performed considerably worse. The results show that model performance under leakage-controlled validation varies considerably across models, and the proposed pipeline offers a transparent, reproducible protocol for evaluating fraud-detection models under severe class imbalance.

Keywords— Credit card fraud detection; data leakage; cross-validation; SMOTE; threshold calibration; focal loss; class imbalance.

I. INTRODUCTION

The increasing sophistication of financial fraud schemes has made the detection of fraudulent transactions increasingly challenging. Online and physical payment channels provide greater convenience but also create opportunities for complex fraudulent activities that can cause substantial financial losses to individuals and organizations [1]–[6]. Consequently, financial

institutions continue to invest in accurate and efficient fraud detection systems capable of identifying suspicious transactions while minimizing financial losses and protecting customers.

Machine learning (ML) has become an important approach for analyzing large-scale transaction data and identifying abnormal patterns associated with fraudulent behavior [7]–

[11]. Unlike conventional statistical approaches, ML techniques can capture complex and nonlinear relationships within financial data and can support the detection of evolving fraud patterns. Classification algorithms, including Logistic Regression, Decision Trees, and Random Forest, have demonstrated their usefulness in identifying irregular transaction patterns and supporting large-scale and real-time fraud detection [12]–[15].

However, severe class imbalance remains one of the main challenges in credit card fraud detection, as legitimate transactions substantially outnumber fraudulent ones [16], [17]. This imbalance can bias learning algorithms toward the majority class and reduce their ability to correctly identify fraudulent transactions. Synthetic Minority Over-sampling Technique (SMOTE) is therefore widely used to generate synthetic minority-class observations and improve the representation of fraudulent transactions during model training [18], [19]. Nevertheless, the effectiveness of such techniques depends strongly on how preprocessing, feature selection, and resampling are integrated into the validation procedure.

A further methodological concern is whether feature selection and resampling are performed independently within each cross-validation fold. When these operations are conducted globally before cross-validation, information from observations that should remain isolated for validation may indirectly influence the training process. The pipeline examined in [20], for example, presents feature selection as part of the general preprocessing stage rather than explicitly repeating it within every cross-validation fold. Similarly, the reported SMOTE-balanced sample size of 226,602 observations per class raises questions about whether resampling was performed independently for each training fold. This study does not conclude that the referenced pipeline is leaky; rather, it identifies an unresolved methodological issue and proposes a stricter validation design in which feature selection, resampling, and threshold calibration are independently performed using only the appropriate training data.

To address this issue, this study develops an iterative and fold-quarantined Knowledge

Discovery in Databases (KDD) pipeline for credit card fraud detection. The proposed approach integrates SelectKBest-based feature selection, SMOTE-based class balancing, and decision-threshold calibration within a stratified k-fold cross-validation framework. The three key contributions of this study are:

Leak-free fold-quarantined architecture: Feature selection and class balancing are performed exclusively on the training portion of each cross-validation fold, ensuring that validation and test observations do not influence the model-development process.

Iterative feature selection within cross-validation: Feature selection is transformed from a one-time global preprocessing operation into an active and iterative data-mining process performed independently within each training fold, allowing the selected features to reflect the information available during each training iteration.

Quarantined threshold calibration: The classification threshold is optimized using a held-out validation subset of the training data rather than relying on the conventional 0.5 threshold, enabling the model to better accommodate severe class imbalance while maintaining an unbiased final evaluation on unseen data.

The proposed framework is designed to provide a more reliable and reproducible evaluation of credit card fraud detection models, particularly under severe class imbalance, by ensuring that preprocessing, feature selection, resampling, and threshold optimization remain isolated from the final evaluation data.

II. RELATED WORK

Recent developments in credit card fraud detection have extensively examined both traditional machine learning and deep learning (DL) frameworks, frequently incorporating strategies to mitigate data imbalance. [2] proposed a hybrid ensemble model combining Iterative Hard Thresholding with Logistic Regression (IHT-LR) and grid search to boost transaction security. Similarly, [1] investigated the robustness of graph neural networks versus camouflaged fraudsters, highlighting the

significance of relational modeling in fraud detection.

To alleviate the problem of imbalanced datasets, [3] proposed quantum machine learning models and showed the potential for using them in this context. [6] and [9] also highlighted the usefulness of supervised learning techniques such as decision trees and logistic regression in identifying transactions. [7] presented a comprehensive review of financial fraud detection frameworks using machine learning and found ensemble methods to be particularly effective.

Previous reviews such as [4] and [13], provide an overview of DL in fraud detection and show that the effectiveness is heavily dependent on feature relevance and model robustness. Additionally, [11] and [15] have shown that ensemble methods such as decision trees, random forests and logistic regression work better than independent models.

Imbalanced learning techniques are another important area of development. [12] and [14] pioneered a real-time fraud detection framework based on ML algorithms and resampling and feature selection techniques. In parallel, previous reviews have shown that data synthesis [21], collaborative learning and hybrid ML-DL techniques [19], [22] can lead to better accuracy and cross-dataset generalizability.

Recent research, such as [10] and [16], investigated the comparison of several algorithms to demonstrate the application of random forest and support vector machines to high-dimensional transaction data. [16], [18] and [23] also highlighted ensemble strategies and anomaly detection techniques for overlapping and minority classes in credit card datasets.

In this paper, we build upon what is already done by putting feature selection and class balancing in a stratified k-fold cross-validation loop rather than applying them equally in all folds before the training. SelectKBest and SMOTE are refitted on their own within the folds using only the training data of the fold to overcome a lack of details in the literature. We evaluate Logistic Regression, Decision Tree, and Random Forest models in the same fold-quarantined process and develop a Multilayer Perceptron with focal loss to compare with the

standard class-weighted loss to identify the key components contributing to the improvement. We also introduce a quarantined threshold calibration step, in which the threshold is set to a held-out validation split only from the training data instead of 0.5. This addresses a second common limitation in the literature where threshold selection is rarely reported in order to improve precision and recall when the class imbalance is high.

III. METHODOLOGY

A. Dataset

The data used in this work is the publicly available Kaggle Credit Card Fraud Detection dataset, which was originally collected and made available by the ULB Machine Learning Group in collaboration with Worldline. It contains 284,807 transactions made by European cardholders over two days in September 2013, of which 492 are fraudulent. All the predictive features (V1–V28) are anonymized using Principal Component Analysis to protect the cardholder's identity as well as Time (seconds since the first transaction) and Amount (which are left untransformed). The binary target Class indicates fraud (1) or genuine (0).

B. Data cleaning

Prior to model training, the raw dataset was cleaned in two steps. First, duplicate transaction records were identified and removed so that the models were trained and evaluated on unique observations and could not benefit from repeated records appearing in both training and test partitions. Second, all 30 input features (V1–V28, Time, and Amount) were checked for missing values. This cleaning was performed once, before cross-validation, because it involves no fitted statistical transformation, no learned parameters, and no use of the target label; it therefore cannot leak information from the test partitions into the training process.

C. Model selection

The choice of models was based on the reference study [20] which evaluated Logistic Regression, Decision Tree, Random Forest, and a deep learning model. We did not repeat a full comparative benchmark for all four but rather

focused on the deep learning model (MLP) to see if the leak-proof, fold-quarantined pipeline can improve its performance compared with a naive implementation. Logistic Regression, Decision Tree, and Random Forest continued to be in the pipeline, so that the performance of the MLP in the pipeline was able to be compared to the baselines and not in isolation.

D. Proposed iterative KDD pipeline

Common practice is to apply feature selection and class balancing globally before the data are split into training and test sets. This can cause data leakage, because properties of the test data inform the fitted transformations, which may inflate performance estimates and reduce generalizability. Fig. 1 shows the fold-quarantined procedure: first, the training fold is split into an inner-train and an inner-validation partition; then feature selection and SMOTE are fitted on the inner-train partition and only applied, never refitted, on the inner-validation and test partitions; finally, the classification threshold is tuned on the inner-validation partition before the test partition is scored.

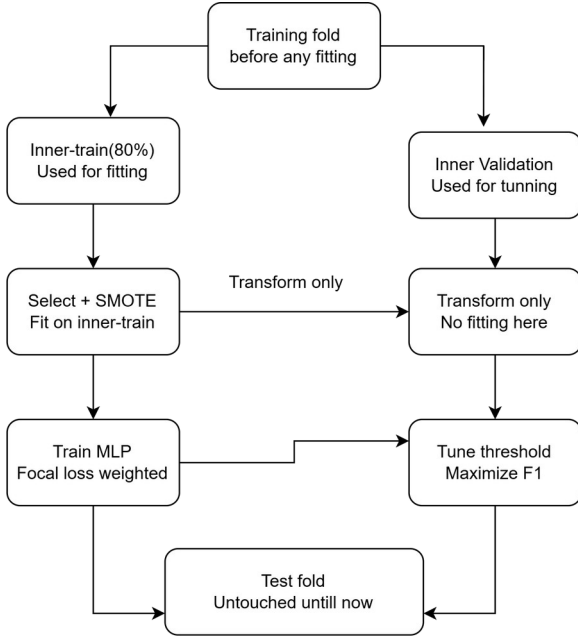


Fig. 1: Fold-quarantined KDD pipeline

To address this issue, we create a fold-quarantined iterative KDD pipeline that uses stratified k-fold cross-validation ($k = 5$) and is designed as follows for each fold:

Inner validation split. The inner validation split was implemented in each training fold to

ensure legitimate and fraudulent cases are represented when tuning the model, rather than a split where one of them is underrepresented in the training data. This split is stratified to keep fraud cases in the training data. It is used only for early stopping and threshold calibration and the test partition is completely quarantined until the final score.

Fold-specific feature selection. Feature selection (SelectKBest, ANOVA F-statistic, $k = 15$) was performed separately within each fold, rather than once on the whole dataset, because the most informative features can differ from one subset of the data to another. Selecting features fold-by-fold ensures that the choice of features reflects only the patterns present in that fold's training data, rather than patterns that might only exist in the specific data the model will later be tested on.

Fold-specific class balancing. SMOTE was applied to address the severe class imbalance in the dataset, since fraudulent transactions represent only 0.17% of all records and a model trained directly on this imbalance tends to be biased toward predicting the majority (legitimate) class. SMOTE was applied only to the training partition of each fold, and not to the test partition, because generating synthetic examples from the test data or from the whole dataset before splitting would let information about the test set influence the training process.

Without this restriction, information from the test data would leak into training and make the model's performance look better than it would be on unseen transactions. Equation (1) generates a new synthetic fraud example by interpolating between a real fraud case x_i and one of its nearest fraud neighbors x_{nn} , using a mixing proportion λ drawn uniformly between 0 and 1. The result is a new data point that lies between the two original cases rather than duplicating either one. Feature scaling (StandardScaler) was likewise fitted only on the training partition of each fold, rather than globally across the full dataset, avoiding even the smaller leakage risk introduced by global scaling.

(1)

E. Model architecture

Table 1 summarizes the configuration used for each baseline model. Default or lightly adjusted hyperparameters were used for these models, as the primary focus of this study is the MLP and its surrounding pipeline rather than exhaustive tuning of the baseline models. The MLP was designed with a progressively narrowing structure (64, 32, 16 units) as shown in Table 2, a common design choice that allows the network to first learn broad feature combinations in the wider early layers, then compress toward more specific, decision-relevant representations in the narrower later layers before the final classification output.

Table 1: Baseline model configuration

Model	Configuration
Logistic Regression	L2 regularization (default), maximum_iteration = 1000 to ensure convergence
Decision Tree	Default splitting criterion (Gini), random_state = 42 for reproducibility
Random Forest	n_estimators = 200, random_state = 42

Table 2: MLP layer configuration

Layer	Units	Activation	Regularization
1	64	ReLU	Batch normalization + Dropout (0.3)
2	32	ReLU	Batch normalization + Dropout (0.3)
3	16	ReLU	
Output	1	Sigmoid	

Batch normalization was applied after the first two dense layers to stabilize training by normalizing layer inputs, reducing sensitivity to weight initialization and allowing a higher effective learning rate. Dropout (rate = 0.3) was applied after the same layers to reduce overfitting, which is a particular risk in this setting given that SMOTE introduces synthetic training examples that are interpolations of a small number of real fraud cases, and a network with excess capacity could otherwise memorize these synthetic patterns rather than learning generalizable fraud indicators. The final hidden layer omits dropout to preserve representational capacity closer to the output. A single-unit

sigmoid output layer was used to produce a probability of fraud, which is then compared against a calibrated threshold rather than a fixed 0.5 cutoff.

The model was trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 256. Early stopping was applied with a patience of 8 epochs, monitoring validation PR-AUC rather than validation loss. PR-AUC was chosen as the monitored metric because, under severe class imbalance, validation loss can continue improving slightly even as the model's ability to distinguish the minority class plateaus or degrades, making PR-AUC a more faithful stopping signal for this task. When training halts, the model's weights are restored to the epoch with the best validation PR-AUC rather than the final epoch. Class weights were computed automatically using scikit-learn's balanced class-weighting scheme and applied on top of the already SMOTE-resampled training data within each fold, providing an additional layer of imbalance correction alongside resampling.

F. Threshold tuning

The default classification threshold of 0.5 assumes that fraud and legitimate cases are roughly equally likely, but our dataset is highly imbalanced, with fraudulent transactions representing only 0.17% of the data. Threshold tuning was therefore introduced to choose a decision threshold better suited to our specific data, by testing many possible threshold values on the inner validation split and selecting the one that performs best on that split, rather than assuming the default 0.5 is appropriate. This threshold, once selected, was applied unchanged when scoring the test partition, ensuring threshold selection itself introduces no leakage.

G. Focal loss ablation

Focal loss (FL) was compared against a standard class-weighted loss function because focal loss is designed to give more weight to examples that are harder to classify correctly, rather than the easy ones, which is relevant given our imbalanced dataset. This comparison was carried out to check whether focal loss provides a meaningful benefit for our specific case, rather than assuming it is necessary simply because it

was used in the reference study. FL (2) reduces the impact of examples the model already classifies confidently and correctly while increasing the impact of examples that it fails to correctly identify.

(2)

H. Experimental setup

The pipeline was implemented in Python 3 and uses scikit-learn for the baseline models, feature selection and evaluation metrics, imbalanced-learn for the SMOTE resampling step and TensorFlow/Keras for building and training the Multilayer Perceptron. All experiments were run on Google Colab using cloud-based GPU acceleration to reduce MLP training time for the different cross-validation folds and ablation configurations. To ensure reproducibility, random seeds were fixed at 42 for NumPy, scikit-learn and TensorFlow components. However, GPU-based neural network training can still exhibit some run-to-run variation even with fixed seeds, due to non-deterministic operations in some GPU computations.

IV. RESULTS AND DISCUSSION

A. Evaluation metrics

Because of the severe class imbalance, raw accuracy was not used as a primary evaluation

metric: a model that labels every transaction as legitimate would still achieve very high accuracy while detecting no fraud. Instead, Precision (3), Recall (4), F1-score (5), Precision-Recall AUC (PR-AUC), and ROC-AUC were measured on the untouched test partition of each fold and reported as mean $\pm$ standard deviation (SD) across folds.

Precision, recall, and F1-score are computed from four possible classification outcomes. A True Positive (TP) is a fraudulent transaction correctly identified as fraud, and a True Negative (TN) is a legitimate transaction correctly identified as legitimate; both are correct predictions. A False Positive (FP) is a legitimate transaction wrongly flagged as fraud, essentially a false alarm, while a False Negative (FN) is a fraudulent transaction that is missed and classified as legitimate, the most costly type of error because the fraud goes undetected.

(3)

(4)

(5)

B. Results

Table 3 reports the mean performance across the five folds for all models evaluated under the proposed fold-quarantined pipeline.

Table 3: Performance metrics across evaluated models (mean across 5 folds)

Model	F1	Precision	Recall	PR-AUC	ROC-AUC
Random Forest	0.843	0.858	0.829	0.839	0.974
MLP (focal loss)	0.815	0.864	0.772	0.794	0.968
MLP (no focal loss)	0.809	0.845	0.776	0.759	0.972
Decision Tree	0.518	0.385	0.793	0.306	0.895
Logistic Regression	0.104	0.055	0.898	0.721	0.977

Random Forest outperformed all other models in this study, likely because it is built as a combination of many decision trees and this ensemble structure works particularly well on tabular data such as this dataset. Instead of a single decision boundary, Random Forest combines the predictions of many trees, which helps it capture patterns in structured, row-and-column financial data more reliably than a single model would.

Logistic Regression had the lowest F1-score of all models. It reached high recall (0.898) but very low precision (0.055), which indicates that, with the default classification threshold, it flagged a very large number of legitimate transactions as fraud. Its PR-AUC (0.721) and ROC-AUC (0.977) are comparatively high, which suggests that the low F1-score largely reflects an uncalibrated threshold rather than a lack of discriminative ability. Because Logistic Regression was evaluated with the default

threshold instead of a tuned one, this comparison is not fully like-for-like with the MLP; applying the same threshold calibration to the baseline models is left to future work.

C. Ablation study

To isolate which part of the pipeline was responsible for the MLP's performance, the model was trained twice under identical conditions: once with focal loss and once with standard class-weighted binary cross-entropy. The results are shown in Table 3.

The small difference between the two configurations ($F1 = 0.815$ versus 0.809) shows that focal loss was not the main driver of the MLP's performance. It did, however, yield a higher precision (0.864 versus 0.845) and a clearer gain in PR-AUC (0.794 versus 0.759), which suggests a modest benefit when the fraud class has very few real samples to learn from. Both configurations share the same pipeline components, namely fold-specific feature selection, SMOTE, class weighting, and calibrated thresholds, so these components, rather than the loss function, probably account for most of the MLP's performance. Isolating their individual contributions would require additional ablations, which are left to future work.

D. Benchmarking to the state-of-the-art

Table 4 compares the F1-scores obtained in this study with those reported in [20]. The comparison is mixed: the proposed pipeline gives gains for some models and losses for others.

Table 4: Comparison with reference study

Model	This study (F1)	Reference study (F1)
Random Forest	0.843	0.744
MLP	0.815	0.889
Decision Tree	0.518	0.909
Logistic Regression	0.104	0.390

Under the proposed pipeline, Random Forest achieved a higher F1-score than reported for Random Forest in the reference study, whereas the MLP, Decision Tree, and Logistic Regression scored lower than the values reported there. These differences should be interpreted

with caution. The pipeline in this study explicitly confines feature selection, resampling, and threshold tuning to the training data of each fold, whereas the reference study does not state whether these steps were performed in that way, so the two sets of results may not have been obtained under equivalent protocols. In addition, some figures reported in the reference study are inconsistent with one another. This is a limitation of the comparison baseline; this study does not attempt to resolve it and reports its own findings independently.

The main contribution of this study is not a higher score across all models but a fully documented, leakage-controlled validation procedure, including quarantined threshold calibration, which the reference study does not describe and which other researchers can adopt to check their own reported results.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a fold-quarantined, iterative KDD pipeline for credit card fraud detection in which feature selection (SelectKBest), class balancing (SMOTE), feature scaling, and decision-threshold calibration are fitted on the training data of each fold only, rather than once before the train-test split. This addresses a methodological gap in [20], where it is not stated whether these steps were repeated within each fold. Under this pipeline, Random Forest achieved the best overall performance ($F1 = 0.843$), about 10 percentage points higher than the Random Forest score reported in Table 3 of [20], while the MLP, Decision Tree, and Logistic Regression scored lower than in [20]. The focal-loss ablation showed only a modest contribution of focal loss (a small gain in precision and PR-AUC). We do not claim that the pipeline in [20] leaked data; rather, the proposed pipeline makes feature selection, class balancing, and threshold calibration explicit, fold-specific, and verifiable, so that other researchers can adopt it and compare their results with ours. A limitation of this study is that no deliberately leaky pipeline was run for comparison, so the size of the leakage effect itself was not quantified.

Future work could further extend this pipeline by testing it with time (chronological) validation

of the dataset's Time column for generalization to strictly future transactions, using the same threshold calibration on models (Logistic Regression and Decision Tree) for fair comparison, further tuning to the MLP architecture and focal loss parameters (α , γ) in the training fold to maintain leak-proof construction, and testing the pipeline on other datasets (such as the PaySim synthetic mobile money dataset in [20]) to confirm that the results generalize beyond the Kaggle dataset. A controlled comparison against a deliberately leaky pipeline (global feature selection and SMOTE before splitting) would also quantify the effect of leakage directly.

DATA AVAILABILITY

The dataset was openly provided by the ULB Machine Learning Group and is available on Kaggle (Credit Card Fraud Detection): <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>.

REFERENCES

- [1] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM international conference on information and knowledge management (CIKM'20)*, ACM, 2020.
- [2] M. A. Talukder, R. Hossen, M. A. Uddin, M. N. Uddin, and U. K. Acharjee, "Securing transactions: a hybrid dependable ensemble machine learning model using IHT-LR and grid search," *Cybersecurity*, vol. 7, p. 32, 2024.
- [3] N. Innan, M. A.-Z. Khan, and M. Bennai, "Financial fraud detection: A comparative study of quantum machine learning models," *Int. J. Quantum Inf.*, vol. 22, no. 02, p. 2350044, Mar. 2024, doi: 10.1142/S0219749923500442.
- [4] Y. Chen, C. Zhao, Y. Xu, C. Nie, and Y. Zhang, "Year-over-Year Developments in Financial Fraud Detection via Deep Learning: A Systematic Literature Review," 2025, *arXiv*. doi: 10.48550/ARXIV.2502.00201.
- [5] T. Guo, Z. Li, and W. Zhang, "Compressive sensing compensation algorithm about nonlinear distortion of clipping for underwater acoustic OFDM systems," *J. Phys. Conf. Ser.*, vol. 1607, no. 1, p. 012038, Aug. 2020, doi: 10.1088/1742-6596/1607/1/012038.
- [6] M. K. H. Chy, "Proactive fraud defense: Machine learning's evolving role in protecting against online fraud," *World J. Adv. Res. Rev.*, vol. 23, no. 3, pp. 1580–1589, Sep. 2024, doi: 10.30574/wjarr.2024.23.3.2811.
- [7] A. Ali *et al.*, "Financial Fraud Detection Based on Machine Learning: A Systematic Literature Review," *Appl. Sci.*, vol. 12, no. 19, p. 9637, Sep. 2022, doi: 10.3390/app12199637.
- [8] B. C. Franco, A. F. B. Costa, and M. A. G. Machado, "Economic-statistical design of the chart used to control a wandering process mean using genetic algorithm," *Expert Syst. Appl.*, vol. 39, no. 17, pp. 12961–12967, Dec. 2012, doi: 10.1016/j.eswa.2012.05.034.
- [9] R. R. Ismail and F. H. Khorsheed, "Classification of Credit Card Frauds Detection using machine learning techniques," *JEECOM J. Electr. Eng. Comput.*, vol. 5, no. 2, pp. 153–160, Oct. 2023, doi: 10.33650/jeeecom.v5i2.6602.
- [10] P. Manorom, U. Detthamrong, and W. Chansanam, "Comparative Assessment of Fraudulent Financial Transactions using the Machine Learning Algorithms Decision Tree, Logistic Regression, Naïve Bayes, K-Nearest Neighbor, and Random Forest," *Eng. Technol. Appl. Sci. Res.*, vol. 14, no. 4, pp. 15676–15680, Aug. 2024, doi: 10.48084/etasr.7774.
- [11] D. Shah and L. Kumar Sharma, "Credit Card Fraud Detection using Decision Tree and Random Forest," *ITM Web Conf.*, vol. 53, p. 02012, 2023, doi: 10.1051/itmconf/20235302012.
- [12] S. K. Hashemi, S. L. Mirtaheri, and S. Greco, "Fraud Detection in Banking Data by Machine Learning Techniques," *IEEE Access*, vol. 11, pp. 3034–3043, 2023, doi: 10.1109/ACCESS.2022.3232287.
- [13] L. Hernandez Aros, L. X. Bustamante Molano, F. Gutierrez-Portela, J. J. Moreno Hernandez, and M. S. Rodríguez Barrero, "Financial fraud detection through the application of machine learning techniques: a literature review," *Humanit. Soc. Sci. Commun.*, vol. 11, no. 1, p. 1130, Sep. 2024, doi: 10.1057/s41599-024-03606-0.
- [14] B. Borketey, "Real-Time Fraud Detection Using Machine Learning," *J. Data Anal. Inf. Process.*, vol. 12, no. 02, pp. 189–209, 2024, doi: 10.4236/jdaip.2024.122011.
- [15] Y. Salunke, S. Phalke, M. Madavi, P. Kumre, and G. Bobhate, "Fraud Detection: A Hybrid Approach With Logistic Regression, Decision Tree, and Random Forest," *Cureus J. Comput. Sci.*, Jan. 2025, doi: 10.7759/s44389-024-02350-5.
- [16] Y. Kumar, S. Saini, and R. Payal, "Comparative Analysis for Fraud Detection Using Logistic Regression, Random Forest and Support Vector Machine," *SSRN Electron. J.*, 2020, doi: 10.2139/ssrn.3751339.
- [17] M. Sopiyan, F. Fauziah, and Y. F. Wijaya, "Fraud Detection Using Random Forest Classifier,

Logistic Regression, and Gradient Boosting Classifier Algorithms on Credit Cards,” *JUITA J. Inform.*, vol. 10, no. 1, p. 77, May 2022, doi: 10.30595/juita.v10i1.12050.

- [18] N. Baisholan, J. E. Dietz, S. Gnatyuk, M. Turdalyuly, E. T. Matson, and K. Baisholanova, “FraudX AI: An Interpretable Machine Learning Framework for Credit Card Fraud Detection on Imbalanced Datasets,” *Computers*, vol. 14, no. 4, p. 120, Mar. 2025, doi: 10.3390/computers14040120.
- [19] E. Btoush, X. Zhou, R. Gururajan, K. C. Chan, and O. Alsodi, “Achieving Excellence in Cyber Fraud Detection: A Hybrid ML+DL Ensemble Approach for Credit Cards,” *Appl. Sci.*, vol. 15, no. 3, p. 1081, Jan. 2025, doi: 10.3390/app15031081.
- [20] T. Albalawi and S. Dardouri, “Enhancing credit card fraud detection using traditional and deep learning models with class imbalance mitigation,” *Front. Artif. Intell.*, vol. 8, p. 1643292, Oct. 2025, doi: 10.3389/frai.2025.1643292.
- [21] A. R. Khalid, N. Owoh, O. Uthmani, M. Ashawa, J. Osamor, and J. Adejoh, “Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach,” *Big Data Cogn. Comput.*, vol. 8, no. 1, p. 6, Jan. 2024, doi: 10.3390/bdcc8010006.
- [22] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: A comparative study,” *Decis. Support Syst.*, vol. 50, no. 3, pp. 602–613, Feb. 2011, doi: 10.1016/j.dss.2010.08.008.
- [23] E. Strelcenia and S. Prakoonwit, “Improving Classification Performance in Credit Card Fraud Detection by Using New Data Augmentation,” *AI*, vol. 4, no. 1, pp. 172–198, Jan. 2023, doi: 10.3390/ai4010008.