Securing Digital Transactions: Machine Learning for Credit Card Fraud Detection

Vinutha H.R
Department of Artificial Intelligence and Data Science
Shridevi Institute of Engineering and Technology
Tumakuru, Karnataka, India
vinuthahr.aids@shrideviengineering.org

Darshan B.A
Department of Artificial Intelligence and Data Science
Shridevi Institute of Engineering and Technology
Tumakuru, Karnataka, India
darshanba8867@gmail.com

Deepak Hosamane M
Department of Artificial Intelligence and Data Science
Shridevi Institute of Engineering and Technology
Tumakuru, Karnataka, India
deepakhosamane01@gmail.com

Dhanush M.S
Department of Artificial Intelligence and Data Science
Shridevi Institute of Engineering and Technology
Tumakuru, Karnataka, India
dhanushms954@gmail.com

Jeevan S.S
Department of Artificial Intelligence and Data Science
Shridevi Institute of Engineering and Technology
Tumakuru, Karnataka, India
jeevanss226@gmail.com

***Abstract*—The rapid growth of digital payments has increased the importance of reliable credit card fraud detection systems. Traditional rule-based approaches can struggle to identify complex and evolving fraud patterns, particularly when fraudulent transactions represent only a small proportion of transaction data. This paper presents a machine learning-based framework that combines data preprocessing, class-imbalance handling, autoencoder-based feature representation, and XGBoost classification. The study uses the publicly available credit card transaction dataset containing 284,807 transactions, including 492 fraudulent transactions. Robust scaling is used to reduce the influence of extreme values, while the Synthetic Minority Oversampling Technique (SMOTE) is applied only to the training data. An autoencoder learns a compact representation of transaction features and calculates reconstruction error; these outputs are combined with the prepared features for downstream classification. XGBoost is used to distinguish legitimate transactions from potentially fraudulent ones. The framework is evaluated using precision, recall, F1-score, area under the precision–recall curve (PR-AUC), and area under the receiver operating characteristic curve (ROC-AUC). The paper also describes an API-oriented deployment workflow. Numerical results should be reported from the final reproducible experiment; illustrative or previously obtained values must not be treated as independently verified without rerunning the implementation.**

***Keywords*—Credit card fraud detection, machine learning, XGBoost, autoencoder, SMOTE, class imbalance, anomaly detection, financial security.**

I. Introduction

The widespread adoption of online shopping, digital banking, and electronic payment systems has significantly increased the volume of credit card transactions. Although these technologies provide convenience and speed, they also create opportunities for fraudulent activities. Credit card fraud can cause financial losses for customers, banks, merchants, and payment service providers. Consequently, effective fraud detection is an important research area in financial cybersecurity.

Traditional fraud detection systems commonly rely on predefined rules, transaction limits, and manually developed patterns. Such systems can identify known suspicious activities but may perform poorly when fraudsters change their behaviour. Rule-based systems can also generate false alarms, resulting in unnecessary transaction declines and additional investigation costs.

Machine learning techniques provide an alternative by learning patterns from historical transaction data. Classification algorithms identify relationships between transaction characteristics and fraud labels, while anomaly detection methods identify observations that differ from learned patterns. However, fraud detection presents several challenges, including extreme class imbalance, changing fraud patterns, and the need for timely predictions.

In commonly used credit card datasets, fraudulent transactions account for only a small fraction of all records. A model that predicts every transaction as legitimate can achieve high accuracy while failing to detect fraud. Therefore, accuracy alone is insufficient. Precision, recall, F1-score, and precision–recall analysis are especially important when evaluating models trained on highly imbalanced data.

This paper presents a framework combining preprocessing, class-imbalance handling, autoencoder-based representation

learning, and XGBoost classification. The goal is to investigate whether learned representations and gradient-boosted trees can support the identification of fraudulent transactions.

II. Motivation and Problem Statement

A. Motivation

Financial institutions process large numbers of transactions every day. Manually investigating every transaction is impractical, and conventional rule-based systems may fail to detect new patterns. An automated system can assist analysts by assigning risk scores and prioritising suspicious transactions for review.

The severe imbalance between legitimate and fraudulent transactions creates an additional challenge. Since fraudulent examples are rare, a classifier may favour the majority class. A reliable experimental design must address imbalance without contaminating the test set and must evaluate minority-class performance directly.

B. Problem Statement

The problem addressed in this research is the identification of fraudulent credit card transactions in a highly imbalanced dataset. The system must prepare transaction features consistently, learn useful patterns, distinguish fraudulent transactions from legitimate transactions, and support efficient inference. The proposed workflow investigates robust scaling, SMOTE, autoencoder-derived features, and XGBoost classification.

C. Research Objectives

The objectives are:

- Preprocess transaction data for machine learning.
- Apply SMOTE exclusively to the training data.
- Learn compact feature representations using an autoencoder.
- Classify transactions using XGBoost.
- Evaluate performance using precision, recall, F1-score, ROC-AUC, and PR-AUC.
- Describe a deployment workflow suitable for transaction-level predictions.

D. Research Contributions

The work presents (i) a modular fraud-detection pipeline, (ii) integration of autoencoder-derived features with a supervised classifier, (iii) an evaluation plan suitable for imbalanced classification, and (iv) an API-oriented design for applying saved preprocessing components and trained models to incoming transactions.

III. Literature Survey

Dal Pozzolo *et al.* discuss practical challenges in credit card fraud detection, including class imbalance, concept drift, and realistic evaluation [**?**]. Their work highlights that fraud detection is not a static classification task because transaction behaviour and fraud strategies can change over time.

Awoyemi *et al.* compare machine learning methods for credit card fraud detection and examine the challenges posed by imbalanced data [4]. Randhawa *et al.* investigate ensemble learning approaches for fraud detection and show the relevance of combining predictive models [5]. Dornadula and Geetha describe a machine learning-based approach for credit card fraud detection [**?**].

Lucas and Jurgovsky review sequence-based fraud detection and discuss the importance of transaction order and changing behaviour [**?**]. These observations suggest that temporal information can be valuable, although a transaction-level model must carefully define which information is available at prediction time.

SMOTE, introduced by Chawla *et al.*, generates synthetic minority-class examples using neighbouring observations [8]. It is commonly used to mitigate imbalance, but it must be applied only within training data to avoid information leakage. Autoencoders learn compressed representations by reconstructing their inputs, and reconstruction error can be used as an additional anomaly-related signal. XGBoost is a scalable gradient-boosting method based on ensembles of decision trees [9].

These studies motivate a pipeline that combines imbalance handling, learned representations, and tree-based classification. Nevertheless, the suitability of the combined approach must be established through reproducible experiments on held-out data rather than assumed from the choice of algorithms.

IV. Proposed System Architecture

The proposed system consists of data ingestion, preprocessing, imbalance handling, autoencoder-based representation learning, XGBoost classification, evaluation, and optional API deployment. Training and inference are separated so that test data do not influence model fitting.

A. System Workflow

The workflow is:

1) Load the transaction dataset and inspect its schema.
2) Check missing values, duplicate records, invalid values, and class distribution.
3) Separate the target label from the input features.
4) Split data into training, validation, and test subsets.
5) Fit the scaler using training data and apply it to the other subsets.
6) Apply SMOTE to the training data only.
7) Train the autoencoder using a documented training strategy.
8) Generate latent features and reconstruction-error features.
9) Train and tune the XGBoost classifier using training and validation data.
10) Evaluate the selected model once on the held-out test set.
11) Save the model and all preprocessing components for inference.

B. Architecture Diagram

Fig. 1 illustrates the main stages. The final implementation should ensure that the same feature ordering and preprocessing transformations are used during training and prediction.

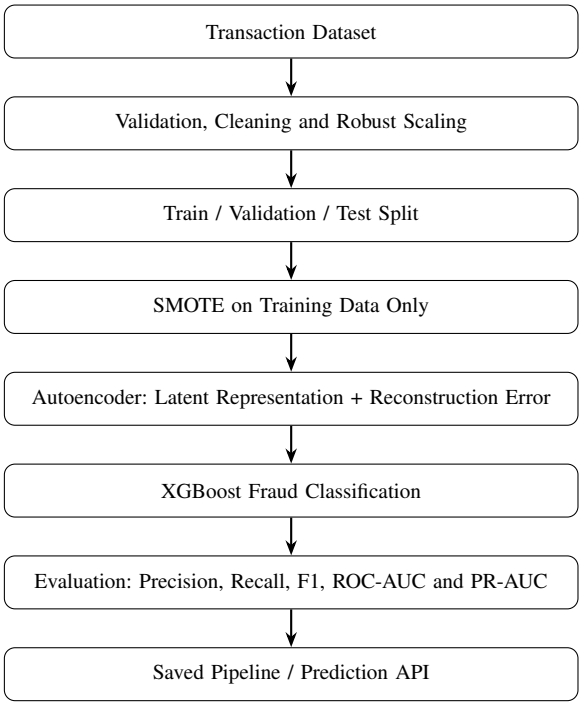


Fig. 1. Proposed credit card fraud detection workflow.

C. Dataset Description

The commonly used public European credit card transaction dataset contains 284,807 transactions and 492 fraud cases. It includes `Time`, `Amount`, anonymised principal component features `V1` through `V28`, and the target column `Class`. A class value of zero represents a legitimate transaction and a value of one represents fraud. The dataset is strongly imbalanced: the fraud proportion is approximately 0.172%.

The data source, download date, exact preprocessing steps, split strategy, software versions, and random seeds should be recorded in the final experiment so that the reported results can be reproduced.

TABLE I
MAIN DATASET FIELDS

Field	Description
Time	Seconds elapsed from the first recorded transaction
Amount	Transaction amount
V1–V28	Anonymised PCA-transformed features
Class	Target label: 0 for legitimate, 1 for fraud

D. Data Preprocessing

The dataset is checked for missing values, duplicate records, and invalid feature values. Any removal or imputation policy should be documented. Extreme observations should not automatically be removed because they may represent genuine fraud or unusual but valid activity.

Robust scaling reduces the influence of extreme values by centring each feature around its median and scaling by its interquartile range:

$$x' = \frac{x - \text{median}(x)}{\text{IQR}(x)}. \tag{1}$$

The median and interquartile range are estimated from training data only. The same fitted transformation is then applied to validation, test, and inference records.

E. Handling Class Imbalance

SMOTE generates synthetic minority-class examples from neighbouring minority observations [8]. In this framework, it is applied after the data split and only to the training set. Validation and test sets retain the naturally observed class distribution. Performing oversampling before splitting could allow information from near-duplicate synthetic examples to cross the split boundary and produce an overly optimistic evaluation.

SMOTE is not guaranteed to improve every classifier. Its effect should be assessed by comparing models with and without resampling, using the same held-out data and evaluation procedure.

F. Autoencoder-Based Representation Learning

An autoencoder learns a compressed representation of its input. The encoder maps the input feature vector x to a latent vector z:

$$z = f_\theta(x), \tag{2}$$

and the decoder reconstructs the input:

$$\hat{x} = g_\phi(z), \tag{3}$$

where f_θ and g_ϕ are the encoder and decoder functions. A common reconstruction objective is mean squared error:

$$E(x) = \frac{1}{d}\sum_{i=1}^{d}(x_i - \hat{x}_i)^2, \tag{4}$$

where d is the number of input features.

The latent representation and reconstruction error can be added to the original prepared features for downstream classification. The training population for the autoencoder must be stated explicitly. If the autoencoder is intended to learn normal behaviour, it should be trained on legitimate training transactions only; if trained on a resampled or mixed-class population, that design choice must be documented. Any fitting or feature selection must avoid use of the test set.

G. XGBoost Classification

XGBoost builds an ensemble of decision trees through gradient boosting [9]. The model score can be expressed as

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \tag{5}$$

where x_i is the input feature vector, f_k is the contribution of tree k, and K is the number of trees. For binary classification, the score is transformed into a probability by the model's logistic objective.

Relevant parameters include the number of estimators, maximum tree depth, learning rate, row subsampling, column subsampling, and regularisation terms. Parameters should be selected using training and validation data. The final test set should not be used repeatedly to select parameters or thresholds.

A probability threshold determines when a transaction is flagged. A lower threshold may increase recall but can also increase false positives. The selected threshold should reflect the intended operating conditions and the cost of missed fraud versus manual review.

V. Implementation and Deployment

A. Implementation Environment

The framework can be implemented in Python using Pandas and NumPy for data handling, scikit-learn for preprocessing and evaluation, imbalanced-learn for SMOTE, XGBoost for classification, and a deep-learning library for the autoencoder. Streamlit can provide an interactive dashboard, while FastAPI can expose a prediction endpoint.

The exact hardware, package versions, training parameters, and random seeds should be reported from the environment used for the final experiment. Hardware descriptions alone do not establish model quality or inference speed.

B. Training Procedure

The target label is separated from the input features before splitting. Stratified splitting can preserve class proportions for an independent and identically distributed evaluation; if the data are ordered in time or the application is intended to predict future transactions, a chronological split may be more appropriate. The chosen strategy must match the intended deployment scenario.

The scaler is fitted only on training observations. SMOTE is then applied to the scaled training features and labels. The autoencoder is trained under the stated strategy, and its latent features and reconstruction errors are generated consistently for the classifier. XGBoost is trained using the resulting training features. Validation data are used for model selection, while the untouched test set is used for final evaluation.

C. Prediction API

A FastAPI service can expose a transaction prediction endpoint. The endpoint validates the input schema, applies the saved scaler, calculates the autoencoder representation and reconstruction error, and calls the saved XGBoost model. It can return a predicted class and a fraud probability, along with a request identifier if the application requires one.

The service should load trained components once at startup rather than retraining for every request. Input validation, structured logging, access control, exception handling, and model-version tracking are important for practical deployment. A model output should be treated as a risk signal to support an appropriate review or decision policy, not as an infallible determination.

D. Prediction Workflow

The inference workflow is:

1) Receive a transaction record.
2) Validate fields and feature dimensions.
3) Apply the saved scaling transformation.
4) Generate the required autoencoder features.
5) Pass the combined feature vector to XGBoost.
6) Apply the configured threshold to the predicted probability.
7) Return the risk score and classification result.

VI. Evaluation Metrics

Because fraud is rare, evaluation should include metrics that reveal minority-class performance. Let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives.

A. Precision

Precision measures the proportion of transactions flagged as fraud that are actually fraudulent:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{6}$$

B. Recall

Recall measures the proportion of all fraudulent transactions that are detected:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{7}$$

C. F1-Score

The F1-score is the harmonic mean of precision and recall:

$$F_1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{8}$$

D. ROC-AUC and PR-AUC

ROC-AUC measures ranking performance across false-positive and true-positive rates at different thresholds. PR-AUC summarizes the precision–recall curve and is especially informative when the positive class is rare. The exact convention used to calculate PR-AUC should be reported because average precision and trapezoidal integration are not identical in all implementations.

E. Confusion Matrix

The confusion matrix gives counts of correct and incorrect predictions. In fraud detection, false negatives represent missed fraud, while false positives represent legitimate transactions flagged for investigation. Both error types matter, but their operational costs may differ substantially.

F. Inference Latency

Latency should be measured with a documented procedure, including hardware, batch size, warm-up behaviour, and whether preprocessing is included. Report the median and a high percentile such as the 95th percentile when possible. A single timing value is not sufficient to establish production performance.

VII. Results and Discussion

Final numerical results must be taken from the actual executed experiment. They should not be invented or inferred from the algorithm names. Table II is provided as a fill-in template. Replace each `TBD` with measured values before submitting the paper.

TABLE II
Test-Set Classification Results

Metric	Measured Value
Precision	TBD
Recall	TBD
F1-score	TBD
ROC-AUC	TBD
PR-AUC / Average Precision	TBD
False positives	TBD
False negatives	TBD
Median inference latency	TBD
95th-percentile latency	TBD

A. Interpreting the Results

Precision should be interpreted as the reliability of alerts, while recall indicates how many fraudulent transactions are captured. A model with high precision but low recall may miss many fraud cases; a model with high recall but low precision may generate a large number of false alarms. The preferred operating point depends on investigation capacity and the relative costs of missed fraud and false positives.

ROC-AUC and PR-AUC should be reported together. On a highly imbalanced dataset, PR-AUC often gives a more direct view of the trade-off between detecting the minority class and keeping alerts relevant. Results should also be compared against a baseline, such as a model without SMOTE or a classifier without autoencoder-derived features, to establish whether each additional component contributes measurable value.

B. Backtesting and Robustness

A single random split may not represent changes in real transaction behaviour. Where timestamps are meaningful, chronological evaluation can better approximate deployment on future transactions. Repeated runs with fixed and documented seeds, threshold sensitivity analysis, and evaluation across different time periods can help assess robustness.

The evaluation should also document the decision threshold, class prevalence in the test set, and whether the reported metrics are calculated from hard class predictions or predicted probabilities. Any inference-time measurement should be made on the saved pipeline rather than only on the classifier in isolation.

VIII. Limitations

The framework has several limitations. First, the public dataset may not represent current fraud strategies or the transaction mix of a specific financial institution. Second, anonymised features limit direct interpretation of individual feature effects. Third, SMOTE can generate synthetic examples that do not fully represent real fraud patterns. Fourth, autoencoder performance depends on the training population, architecture, and optimisation settings. Fifth, XGBoost predictions may degrade when the underlying data distribution changes.

The framework also does not, by itself, model all operational constraints such as transaction authorisation rules, investigation capacity, customer impact, regulatory requirements, privacy controls, or the cost of false declines. A deployment decision requires broader validation, monitoring, and governance.

IX. Future Scope

Future work can evaluate the model on chronological and institution-specific datasets, compare XGBoost with other ensemble and deep-learning methods, and study cost-sensitive learning or alternative imbalance-handling methods. Temporal models may be explored when reliable transaction sequences are available. Calibration, drift detection, threshold adaptation, explainability methods, and privacy-preserving training are also potential extensions.

A production-oriented system could add monitoring dashboards, model-version tracking, secure logging, analyst feedback, and scheduled retraining. Such extensions should be tested carefully to ensure that new information does not leak into historical evaluation and that model changes are traceable.

X. Conclusion

This paper described a machine learning framework for credit card fraud detection that combines robust preprocessing, training-only SMOTE, autoencoder-based representation learning, and XGBoost classification. The framework emphasises evaluation metrics suited to imbalanced data, including precision, recall, F1-score, ROC-AUC, and PR-AUC. It also outlines a deployment workflow in which saved preprocessing components and trained models are used to score new transactions.

The proposed architecture provides a structured basis for experimentation, but its effectiveness must be demonstrated using measured results on an untouched test set and, ideally,

on data from later time periods. Reproducible reporting of data preparation, training configuration, decision threshold, metrics, and latency is essential before drawing conclusions about practical fraud-detection performance.

REFERENCES

[1] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, 2018, doi: 10.1109/TNNLS.2017.2736643.

[2] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating Probability with Undersampling for Unbalanced Classification," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2015, pp. 159–166.

[3] A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, S. Waterschoot, and G. Bontempi, "Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4915–4928, 2014, doi: 10.1016/j.eswa.2014.02.026.

[4] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare, "Credit Card Fraud Detection Using Machine Learning Techniques: A Comparative Analysis," in *Proceedings of the International Conference on Computing Networking and Informatics*, 2017, pp. 1–9.

[5] K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi, "Credit Card Fraud Detection Using AdaBoost and Majority Voting," *IEEE Access*, vol. 6, pp. 14277–14284, 2018, doi: 10.1109/ACCESS.2018.2806420.

[6] V. N. Dornadula and S. Geetha, "Credit Card Fraud Detection Using Machine Learning Algorithms," *Procedia Computer Science*, vol. 165, pp. 631–641, 2020, doi: 10.1016/j.procs.2020.01.057.

[7] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, and O. Caelen, "Sequence Classification for Credit-Card Fraud Detection," *Expert Systems with Applications*, vol. 100, pp. 234–245, 2018, doi: 10.1016/j.eswa.2018.01.037.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002, doi: 10.1613/jair.953.

[9] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006, doi: 10.1126/science.1127647.

[11] J. Davis and M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 233–240, doi: 10.1145/1143844.1143874.

[12] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006, doi: 10.1016/j.patrec.2005.10.010.

[13] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009, doi: 10.1109/TKDE.2008.239.

[14] R. Bin Sulaiman, V. Schetinin, and P. Sant, "Review of Machine Learning Approach on Credit Card Fraud Detection," *Human-Centric Intelligent Systems*, vol. 2, pp. 55–68, 2022, doi: 10.1007/s44230-022-00004-0.

[15] E. A. L. M. Btoush, X. Zhou, R. Gururajan, K. C. Chan, R. Genrich, and P. Sankaran, "A Systematic Review of Literature on Credit Card Cyber Fraud Detection Using Machine and Deep Learning," *PeerJ Computer Science*, vol. 9, Art. no. e1278, 2023, doi: 10.7717/peerj-cs.1278.

[16] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer, and G. Bontempi, "SCARFF: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark," *Information Fusion*, vol. 41, pp. 182–194, 2018, doi: 10.1016/j.inffus.2017.09.005.