IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

AI-Powered Real-Time Financial Fraud Detection System

Shreyash Karpe[1], Kalash Jain[2], Amrut Shelke[3], Pavan Tank[3]
Department of Computer Science and Engineering[1-3]
MIT School of Computing, MIT Art, Design and Technology University, Pune, India
[3]amrutshelke1210@gmail.com

Abstract : *Digital payment systems need to flag potentially fraudulent transactions while keeping alert volumes manageable for analysts. This paper presents a prototype that processes structured transaction data, derives behavioral features, and uses supervised classifiers to assign a continuous fraud risk score to each transaction. Configurable thresholds map scores to three categories: Genuine, Suspicious, and High-Risk. A React dashboard displays transaction scores, prioritizes alerts, and can show the features that contributed to a score. The evaluation design compares logistic regression, random forest, and XGBoost with a rule-based baseline using public credit card data and synthetic transactions. Illustrative scores demonstrate the comparison across classification and ranking metrics. The paper describes the serving architecture, the latency reported for the prototype, and the additional testing required to assess detection performance and latency under operational conditions.*

Keywords: Financial fraud detection, machine learning, XGBoost, real-time inference, risk scoring, explainable AI, class imbalance, analyst dashboard.

I. INTRODUCTION

Digital wallets, online banking, and instant payments have increased the volume and speed of electronic transactions. Static rule-based fraud detection systems are easy to audit, but maintaining them requires continual manual effort, and fixed rules may miss changes in transaction behavior or create excessive alerts [1]. Machine learning can learn from labeled transactions, while an operational system also needs fast scoring, manageable alert volumes, and explanations that support analyst review.

This paper describes a prototype that separates offline model training from online scoring, uses configurable thresholds to turn model scores into review categories, and integrates an analyst dashboard. It outlines a protocol for comparing classifiers with a rule-based baseline and discusses the evidence needed to evaluate fraud detection and scoring latency.

II. METHODOLOGY

A. System Architecture

The presentation layer is a React single-page application that displays a live transaction feed, score distributions, alert queues, and transaction-level explanations. A Python API validates requests, computes features, invokes the model, applies thresholds, and persists the results. The model layer contains a serialized classifier, the shared feature pipeline, and an optional SHAP explainer. The data layer stores transactions, engineered features, risk scores, and alert status. Offline training and evaluation take place outside the serving path.

B. Data and Feature Engineering

The evaluation design uses the public European credit card fraud dataset [2] and synthetic transactions with additional fields such as merchant category, payment channel, device type, and coarse location. Where inputs contain the required identifiers and history, the feature pipeline derives transaction counts over time windows, rolling amount statistics, and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, November 2026 Impact Factor: 8.2

deviations from historical geographic and channel patterns. Training and serving use the same transformation pipeline. Class weighting and limited oversampling are applied within the training folds.

C. Models and Evaluation Protocol

The evaluation design uses stratified training, validation, and test partitions and stratified cross-validation for logistic regression, random forest, and XGBoost. The classifiers are compared with a rule-based baseline. Evaluation metrics include precision, recall, F1-score, area under the receiver operating characteristic curve (ROC AUC), and average precision (AP). The latency measures are the median and 95th-percentile scoring times under concurrent load.

The classifier produces a risk score from 0 to 1. Configurable thresholds map it to Genuine, Suspicious, or High-Risk. High-Risk transactions trigger dashboard alerts. The precision–recall curve and analyst review capacity inform threshold selection. Probability calibration should be evaluated before interpreting scores as probabilities, especially when sampling alters class proportions [3].

D. Implementation

The prototype pairs a React dashboard with a Python service and uses scikit-learn, XGBoost, imbalanced-learn, SHAP, and joblib. The design allows FastAPI or Flask for the service and MongoDB or PostgreSQL for persistence. Keeping the service stateless supports replication, with stored transaction and alert data handled by the data layer. The shared feature module and stable API contract allow model replacement without changes to the dashboard.

The dashboard summarizes scores and risk categories over time, lists alert details, and can request the features that most influenced a high-risk score. Authentication protects access to the service and dashboard. The project describes unit tests for feature computation and threshold mapping and integration tests covering request handling, scoring, and persistence.

III. LITERATURE REVIEW

Research on transaction features shows how historical behavior can inform credit card fraud detection [4]. XGBoost provides a scalable implementation of gradient-boosted trees [5]. Class imbalance motivates methods such as SMOTE and cost-sensitive learning [6], along with care in how scores are calibrated and how operating thresholds are set [3].

Carcillo et al. describe a scalable streaming framework for credit card fraud detection [7]. Jurgovsky et al. model transaction sequences to incorporate temporal behavior [8]. The prototype uses transaction-level ensemble models and an optional SHAP explanation to show analysts which input features drove a score [9]. The implementation uses scikit-learn for model development [10].

IV. RESULTS AND DISCUSSION

Table I reproduces the illustrative values from the project documentation. In this example, XGBoost ranks highest on every metric, followed by random forest. The values illustrate the reporting format and the trade-off between precision and recall. Estimating the performance of each model requires reproducible held-out test results.

TABLE I. ILLUSTRATIVE COMPARISON OF FRAUD DETECTION MODELS

Model	Precision	Recall	F1-score	ROC AUC	AP
Logistic regression	0.72	0.65	0.68	0.91	0.74
Random forest	0.85	0.78	0.81	0.96	0.86
XGBoost	0.88	0.82	0.85	0.97	0.89
Rule-based baseline	0.45	0.55	0.50	0.70	0.48

The project documentation reports median scoring latency below 150 ms and 95th-percentile latency below 250 ms on commodity hardware under moderate concurrency. The reported path covers feature computation and prediction. Detailed hardware specifications, request rates, and benchmark samples are needed to reproduce these figures or extrapolate to production volumes.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, November 2026 Impact Factor: 8.2

The alert categories show how an analyst can use model scores. For example, a large transaction from an unusual location during a burst of activity may score in the High-Risk range, prompting closer review. A transaction consistent with prior behavior may fall in the Genuine range. In deployment, the alert threshold affects analyst workload and the number of fraud cases detected. The illustrative scores alone cannot establish an operating threshold.

Synthetic data can demonstrate the use of attributes omitted from the public benchmark, but the mix of synthetic and public data, feature availability, and thresholds affect the interpretation of results. The evaluation also leaves open how well performance carries over to changing transaction patterns, delayed labels, and coordinated fraud across accounts.

V. CONCLUSION

The prototype combines a shared feature pipeline, ensemble classification, configurable risk categories, and a React dashboard for analyst review. The illustrative comparison demonstrates how to report detection and ranking metrics, and the documented latency figures provide a target for repeatable load testing. Further work includes validating performance on held-out transactions, documenting dataset composition and scoring conditions, and evaluating drift monitoring, analyst feedback, fairness, and models that capture sequential or cross-account activity.

ACKNOWLEDGMENTS

The authors acknowledge Worldline and the Machine Learning Group at Université Libre de Bruxelles for making the public credit card fraud dataset available [2].

DATA ARCHIVING STATEMENT

The public credit card fraud dataset is hosted in the ULB Machine Learning Group's Kaggle repository [2]. The manuscript describes an additional synthetic transaction set for which no public archive is identified.

REFERENCES

[1] J. West and M. Bhattacharya, "Intelligent financial fraud detection: A comprehensive review," Comput. Secur., vol. 57, pp. 47–66, 2016.

[2] Machine Learning Group, Université Libre de Bruxelles, "Credit Card Fraud Detection," Kaggle dataset. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

[3] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in Proc. IEEE Symp. Series Comput. Intell., 2015, doi: 10.1109/SSCI.2015.33.

[4] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature engineering strategies for credit card fraud detection," Expert Syst. Appl., vol. 51, pp. 134–142, 2016.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 785–794.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[7] F. Carcillo et al., "SCARFF: A scalable framework for streaming credit card fraud detection with Spark," Inf. Fusion, vol. 41, pp. 182–194, 2018.

[8] J. Jurgovsky et al., "Sequence classification for credit-card fraud detection," Expert Syst. Appl., vol. 100, pp. 234–245, 2018.

[9] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Adv. Neural Inf. Process. Syst., vol. 30, 2017.

[10] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.