

An Explainable Machine Learning Framework for Real-Time Credit Card Fraud Detection using Random Forest and SHAP

Payal Rajendra Thube Department of Master of Computer Applications Parvattibai Genba Moze College of Engineering Savitribai Phule Pune University, Wagholi, Pune, India Email: thubepayal@gmail.com

Prof. Raviraj Bhairat (Guide) Department of Master of Computer Applications Parvattibai Genba Moze College of Engineering Savitribai Phule Pune University, Wagholi, Pune, India

ABSTRACT

The exponential growth of digital payments has been accompanied by a sharp rise in credit card fraud, exposing financial institutions and consumers to large monetary losses. Traditional rule-based fraud detection systems suffer from high false-positive rates, an inability to adapt to evolving fraud patterns, and a lack of transparency in their decisions. This paper presents an explainable, end-to-end machine learning framework for real-time credit card fraud detection. The proposed system integrates a Random Forest classifier with SHapley Additive exPlanations (SHAP) to deliver both accurate and interpretable predictions. The framework is implemented as a three-tier web application using FastAPI for the backend, Streamlit for an interactive frontend, PostgreSQL for persistent storage, and Redis for caching and session management. A built-in transaction simulator, a three-strike account-blocking mechanism, and an SMTP-based alert subsystem are introduced to demonstrate operational behavior under continuous load. Experimental evaluation shows that the model achieves over 93% classification accuracy with sub-100 ms single-prediction latency, while SHAP feature attributions provide analyst-friendly justifications for every decision. The results demonstrate that high accuracy, low latency, and transparent decision-making can be jointly achieved within a single, deployable architecture suitable for fintech and banking environments.

Keywords: Credit card fraud detection, Machine learning, Random Forest, Explainable AI, SHAP, Real-time systems, FastAPI, Streamlit, FinTech.

I. INTRODUCTION

Online and card-based transactions have become the dominant medium of financial exchange in the modern digital economy. According to the Reserve Bank of India's annual report on banking frauds, the volume and value of card and internet banking frauds have grown consistently over the last decade [6]. As fraudsters adopt increasingly sophisticated techniques, conventional rule-based detection mechanisms — which rely on static thresholds (e.g., transaction amount limits or velocity rules) — are no longer sufficient. Such systems exhibit three recurring weaknesses: (i) high false-positive rates that erode customer trust, (ii) inability to generalize to previously unseen fraud patterns, and (iii) opacity of decisions, which complicates regulatory audits and analyst review.

Machine learning (ML) offers a natural alternative because models can learn complex, non-linear patterns from historical transaction data and adapt to new behavior through retraining. However, the deployment

of ML in financial systems introduces its own set of challenges. Production fraud detection requires sub-second inference latency, robust authentication and authorization, automated alerting, and — critically — the ability to *explain* each prediction. Regulators and analysts must be able to understand why a transaction was flagged before the system can be trusted to block accounts or initiate disputes.

This paper addresses these requirements by proposing an explainable, real-time fraud detection framework with the following contributions:

- A Random Forest-based fraud classification pipeline trained on engineered transaction features, achieving accuracy above 93% with sub-100 ms inference latency.
- Integration of SHAP [1, 2] to produce per-transaction feature attributions, exposed directly through the web interface.
- A complete three-tier deployable architecture using FastAPI, Streamlit, PostgreSQL, and Redis, demonstrating that ML, security, and

explainability can be combined in a single coherent system.

- An operational layer comprising a transaction simulator, a three-strike account-blocking mechanism, and SMTP-based fraud alerts, supporting realistic evaluation and demonstration.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the proposed system architecture. Section IV details the methodology, including feature engineering, modeling, and explainability. Section V presents the implementation. Section VI reports experimental results. Section VII discusses limitations, Section VIII outlines future work, and Section IX concludes.

II. RELATED WORK

Research on credit card fraud detection has progressed from statistical approaches and hand-crafted rules to a broad spectrum of data-driven methods. Bhattacharyya *et al.* [3] provided a comparative study of data-mining techniques on real-world credit card transactions, evaluating logistic regression, neural networks, and support vector machines. Their work highlighted the severe class imbalance inherent in fraud data and motivated the use of sampling and cost-sensitive techniques.

Ensemble methods, particularly Random Forests [4], have repeatedly been shown to be effective for tabular fraud detection data because of their robustness to noise, native handling of mixed feature types, and resistance to overfitting. Dal Pozzolo *et al.* [5] addressed the calibration of fraud classifiers under undersampling, an important consideration for deployments where output probabilities are used to set alert thresholds.

More recent work has explored deep architectures such as LSTMs, autoencoders, and Graph Neural Networks (GNNs) for sequential and relational fraud detection, and has emphasized the need for *explainability* in financial AI systems. SHAP [1, 2] has emerged as a unifying framework for model-agnostic feature attribution and is particularly attractive for tree-based models due to the existence of efficient, exact computation through TreeSHAP.

Despite this rich body of work, a gap remains in publicly documented, end-to-end systems that integrate accurate ML, model explainability, secure user-facing

deployment, and operational tooling such as alerting and account control. The framework presented in this paper targets exactly this gap.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed Credit Card Fraud Detection (CCFD) system follows a classical three-tier architecture, instantiated with modern Python web technologies.

Figure 1 – System Architecture: Streamlit Frontend → FastAPI Backend → {ML Engine, PostgreSQL, Redis, SMTP}.

A. Presentation Layer

The presentation layer is implemented in Streamlit, a Python framework for rapidly building data applications. It exposes the following interfaces to the user:

- **Authentication pages:** Registration, login, and password reset flows.
- **Single-prediction page:** A form for entering transaction details (amount, type, merchant category, hour of day, day of week, foreign transaction flag) and receiving a fraud/authentic decision with the associated probability and SHAP explanation.
- **Bulk upload page:** A CSV-upload interface for batch scoring with row-level error reporting.
- **Analytics dashboards:** ROC curves, confusion matrices, feature-importance charts, and category-wise breakdowns.
- **Transaction simulator:** A controllable engine that generates realistic transactions to demonstrate continuous monitoring.

B. Application Layer

The application layer is built with FastAPI and structured around clearly separated routers for authentication, prediction, history, analytics, and simulation control. JSON Web Tokens (JWT) are used for stateless authentication; password hashes are computed with bcrypt. The ML inference module loads a serialized Random Forest model and a fitted preprocessing pipeline at startup, yielding sub-100 ms single-prediction latency.

C. Data Layer

PostgreSQL serves as the primary relational store, with tables for users, transactions, simulator customers, and

fraud alerts (see Section V). Redis is used for caching feature aggregates and for session management, reducing repeated database hits during high-volume prediction bursts. SMTP (via Gmail) is used as the alert channel; the fraud-alert subsystem implements a three-strike rule that automatically blocks accounts after three consecutive flagged transactions.

IV. METHODOLOGY

A. Problem Formulation

Let $\mathbf{x} \in \mathbb{R}^d$ denote the feature vector representing a single transaction, and let $y \in \{0, 1\}$ be the fraud label, where $y = 1$ denotes a fraudulent transaction. The objective is to learn a function $f: \mathbb{R}^d \rightarrow [0, 1]$ such that $f(\mathbf{x})$ approximates $P(y = 1 | \mathbf{x})$. A threshold $\tau \in [0, 1]$ is then applied to obtain the discrete label.

B. Feature Engineering

The system uses the following families of features, derived from raw transaction records:

- **Monetary features:** Transaction amount and log-transformed amount.
- **Categorical features:** Transaction type (Online, POS, ATM, Fund Transfer), merchant category (Retail, Travel, Entertainment, Healthcare, etc.), and a foreign-transaction flag, encoded via one-hot encoding.
- **Temporal features:** Hour of day (0–23) and day of week (0–6), capturing periodic transaction behavior.
- **Behavioral features:** User-level rolling statistics such as average and maximum transaction amount over recent windows and the count of high-value transactions in the last 24 hours, computed at inference time using a Redis-backed cache.

C. Model Selection: Random Forest

A Random Forest classifier [4] was selected for its strong performance on tabular financial data, native handling of mixed feature types, robustness to outliers, and the availability of an exact and efficient SHAP implementation (TreeSHAP). The model is trained with:

- Number of trees: $T = 200$
- Maximum tree depth: tuned via grid search, typically $d_{\max} \in [10, 20]$

- Class weighting: balanced to mitigate class imbalance
- Out-of-bag scoring enabled for unbiased generalization estimates

The dataset is split into 70% training, 15% validation, and 15% test partitions in a stratified fashion to preserve fraud prevalence in each split.

D. Explainability with SHAP

For every prediction, the system computes Shapley values $\phi_i(\mathbf{x})$ for each feature i using TreeSHAP, satisfying:

$$f(\mathbf{x}) = \phi_0 + \sum_{i=1}^d \phi_i(\mathbf{x})$$

where ϕ_0 is the expected model output. The top-ranked features by absolute Shapley value are displayed alongside the fraud/authentic decision, enabling analysts to audit each prediction. For example, an alert may be accompanied by an explanation such as “high amount (+0.21), foreign transaction (+0.14), unusual hour (+0.09)”.

E. Fraud Alerting and Account Control

On a positive prediction (i.e., $f(\mathbf{x}) \geq \tau$), the system:

1. Persists the flagged transaction to the database with its SHAP explanation.
2. Increments a per-user fraud counter.
3. Sends an SMTP email alert to the administrator and the user.
4. If the counter reaches three within the configured window, the account is automatically blocked and an audit record is written.

V. IMPLEMENTATION

A. Technology Stack

Layer	Technology
Frontend	Streamlit, HTML/CSS
Backend	FastAPI, SQLAlchemy, Python-JOSE (JWT)
ML Engine	scikit-learn, SHAP, Pandas, NumPy
Database	PostgreSQL 12+
Cache	/ Redis 6+
Session	
Alerts	SMTP (Gmail)
Server	Uvicorn (ASGI)

B. Database Schema

Four core relations are maintained: users, transactions, simulator_customers, and fraud_alerts. The transactions table stores the amount, type, merchant category, timestamp, ground-truth label (when known), the model's prediction is_fraud, and a JSON-encoded SHAP explanation, supporting both online inference and offline analysis.

C. API Design

The FastAPI backend exposes RESTful endpoints, including:

- POST /auth/register, POST /auth/login
- POST /predict — single-transaction scoring
- POST /upload — bulk CSV scoring
- GET /history — paginated transaction history
- GET /analytics — aggregate metrics for the dashboards
- POST /simulator/start, POST /simulator/stop — simulator control

Authentication is enforced through a dependency-injected JWT verifier; unauthenticated requests are rejected with HTTP 401.

D. Transaction Simulator

To emulate continuous monitoring, a transaction simulator generates synthetic transactions for a configurable population of customers at adjustable rates. Each generated transaction is scored end-to-end through the same prediction pipeline, exercising the alerting and blocking logic and producing a live feed in the user interface.

VI. EXPERIMENTAL EVALUATION

A. Experimental Setup

Experiments were conducted on a workstation with an Intel Core i5 processor, 8 GB RAM, running Ubuntu 20.04. Models were trained using scikit-learn 1.x. Latency was measured end-to-end from HTTP request to JSON response, averaged over 1,000 single-prediction calls.

B. Evaluation Metrics

Given the imbalanced nature of fraud data, accuracy alone is not sufficient. The following metrics were computed on the held-out test set:

- **Accuracy:** $(TP + TN) / (TP + TN + FP + FN)$
- **Precision:** $TP / (TP + FP)$
- **Recall (Sensitivity):** $TP / (TP + FN)$
- **F1-score:** harmonic mean of precision and recall
- **ROC-AUC:** area under the receiver operating characteristic curve

C. Classification Performance

Metric	Value
Accuracy	0.94
Precision	0.92
Recall	0.90
F1-score	0.91
ROC-AUC	0.96

The model exceeds the 93% accuracy target stated in the system objectives and achieves a strong ROC-AUC, indicating robust separability between fraudulent and legitimate transactions.

D. Latency

Single-transaction prediction latency, including network round-trip, ORM access, model inference, and SHAP computation, averaged **~80 ms**, comfortably below the 100 ms non-functional target. Bulk processing through the CSV upload endpoint sustained several hundred predictions per second on the same hardware.

E. Explainability Evaluation

Qualitative inspection of SHAP explanations confirmed that the most influential features for fraud predictions were, in decreasing order: transaction amount, merchant category, foreign-transaction flag, and hour of day. These attributions are consistent with prior empirical findings and with domain intuition, supporting the trustworthiness of the model.

F. Operational Behavior

Stress testing through the transaction simulator showed that the system correctly:

- Sent SMTP fraud alerts within seconds of a positive prediction.
- Incremented per-user fraud counters and triggered automatic account blocking after three flagged transactions.
- Maintained UI responsiveness under continuous transaction streams thanks to Redis caching of recent activity.

VII. DISCUSSION AND LIMITATIONS

While the proposed framework achieves its stated objectives, several limitations should be acknowledged. First, the system relies on a supervised model and therefore inherits the well-known difficulty of *concept drift*: as fraudsters change tactics, model performance will degrade until retraining. Second, the experimental evaluation focuses on a single dataset; cross-bank generalization has not been validated. Third, although SHAP attributions improve transparency, they are local explanations and do not by themselves guarantee fairness or absence of bias. Fourth, full regulatory readiness (PCI-DSS, GDPR) requires production-grade hardening, audit logging, and data residency controls beyond the scope of this academic prototype. Finally, the alerting subsystem currently uses email only; production deployments would integrate SMS, push, and case-management systems.

VIII. FUTURE WORK

Several extensions are planned:

- **Concept drift adaptation:** Online learning techniques and drift detectors with automated retraining.
 - **Deep sequential models:** LSTM- and Transformer-based architectures to capture temporal patterns within a customer's transaction sequence.
 - **Graph Neural Networks:** Modeling customer–merchant–device relationships to detect fraud rings.
 - **Federated learning:** Multi-institution training without sharing raw customer data.
 - **Streaming integration:** Connection to bank transaction streams via Kafka, replacing the simulator with live feeds.
 - **Containerization:** Migration to Docker and Kubernetes for horizontal scalability and zero-downtime deployments.
-

IX. CONCLUSION

This paper presented an explainable, real-time credit card fraud detection framework that combines a Random Forest classifier with SHAP explanations and an operational web stack composed of FastAPI,

Streamlit, PostgreSQL, and Redis. The system meets demanding non-functional targets — accuracy above 93% and sub-100 ms inference latency — while making each decision auditable through per-feature Shapley attributions. The integration of an alerting subsystem, three-strike account blocking, and a transaction simulator demonstrates that academic ML prototypes can be designed with production realism in mind. The framework can serve as a foundation for future research in concept drift handling, deep sequence models, graph-based fraud detection, and federated learning across financial institutions.

ACKNOWLEDGMENT

The author thanks Prof. Raviraj Bhairat for his guidance throughout this project, the faculty of the Department of Master of Computer Applications at Parvattibai Genba Moze College of Engineering, Wagholi, Pune, and Shorat Innovations Pvt. Ltd. for their support during the associated industrial internship.

REFERENCES

- [1] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Proc. 31st Int. Conf. Neural Information Processing Systems (NIPS)*, 2017, pp. 4765–4774.
- [2] S. M. Lundberg *et al.*, “From local explanations to global understanding with explainable AI for trees,” *Nature Machine Intelligence*, vol. 2, no. 1, pp. 56–67, 2020.
- [3] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: A comparative study,” *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.
- [4] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [5] A. Dal Pozzolo, O. Caen, R. A. Johnson, and G. Bontempi, “Calibrating probability with undersampling for unbalanced classification,” in *Proc. IEEE Symp. Computational Intelligence and Data Mining (CIDM)*, 2015.
- [6] Reserve Bank of India, “Annual report on banking frauds in India,” 2023. [Online]. Available: <https://www.rbi.org.in/>

[7] S. Ramírez, “FastAPI — Modern, fast web framework for building APIs.” [Online]. Available: <https://fastapi.tiangolo.com/>

[8] Streamlit Inc., “Streamlit — The fastest way to build data applications.” [Online]. Available: <https://streamlit.io/>

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

[10] SHAP Contributors, “SHAP (SHapley Additive exPlanations) documentation.” [Online]. Available: <https://shap.readthedocs.io/>

[11] PostgreSQL Global Development Group, “PostgreSQL: The world’s most advanced open source database.” [Online]. Available: <https://www.postgresql.org/>

[12] Redis Ltd., “Redis — The real-time data platform.” [Online]. Available: <https://redis.io/>