



Real-Time UPI Fraud Detection Using a Hybrid XG Boost-LSTM Ensemble with Integrated Explainability

Sayeeda Arshiya¹, Dr. Manjula R. Chougala²

¹M Tech Scholar, Computer Science and Engineering, M S Ramaiah Institute of Technology Bangalore, India.

²Assistant Professor, Computer Science and Engineering, M S Ramaiah Institute of Technology Bangalore, India.

To Cite this Article: Sayeeda Arshiya¹, Dr. Manjula R. Chougala², "Real-Time UPI Fraud Detection Using a Hybrid XG Boost-LSTM Ensemble with Integrated Explainability", Indian Journal of Computer Science and Technology, Volume 05, Issue 02 (May-August 2026), PP: 985-989.



Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by-nc-nd/4.0/); Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: Each month, more than 12 billion payments flow through India's UPI system. This digital network now stands at the center of everyday money transfers across the country.. However, the instantaneous settlement characteristic of UPI shrinks the fraud-intervention window to under two seconds, rendering conventional rule-based filters insufficient. This paper presents a production-oriented fraud detection framework that fuses gradient-boosted decision trees (GradientBoosting/XGBoost-equivalent) for static tabular feature analysis with a custom LSTM network for sequential behavioural pattern modelling. Fewer examples of one class get balanced by a tailored method lifting its share from 5% to about 39% during training. One part leans heavier when merging results - set at 0.6 - to shape the final fraud likelihood. Messages flow through a Flask setup built like Kafka, answering most requests under 25 milliseconds even with over a thousand each second. Integrated SHAP global explanations and LIME per-case explanations satisfy emerging regulatory requirements for transparent automated financial decisions. A test using 20,000 made-up UPI transactions across five types of attacks shows the mixed model works best. It hits an F1-score of 0.958 and a ROC-AUC of 0.9985. Better results than each of the four comparison models when looking at both spotting threats and real-world use. Security for data transfer relies on HTTPS/TLS plus identity checks through tokens. Each request must carry a valid access key to proceed.

Keywords: UPI fraud detection; XGBoost; LSTM; ensemble learning; SMOTE; SHAP; LIME; real-time inference; explainable AI; digital payments.

I. INTRODUCTION

Backed by India's NPCI, the Unified Payments Interface started in April 2016. By late 2024, it handles over 12 billion transactions every month. Transaction values climb past ₹18 trillion each month. Speed matters here

- most payments settle before two seconds pass. Designed for smooth connections between systems, the process moves fast: tap, confirm, done. Credit shows almost instantly. That same architectural imperative, however, eliminates the grace period that traditional fraud teams rely on to quarantine suspicious transfers. Once a fraudulent transaction clears the authentication gate, funds reach the beneficiary account almost instantly and recovery becomes operationally and legally complex.

The threat landscape facing UPI encompasses five principal attack families. QR-code reversal encodes a debit request within a code that appears to initiate a credit, exploiting the visual indistinguishability of collect and pay flows from the user's perspective. SIM-swap attacks persuade mobile network operators to port the victim's phone number to an attacker-controlled SIM, hijacking the one-time password channel. Fake collect requests impersonate cashback or refund flows to trick users into authorising unsolicited debits. Credential phishing captures UPI PINs through social engineering or screen-sharing applications. Account-takeover attacks compromise third-party UPI applications to gain full programmatic control over linked accounts [3].

Rule-based fraud filters, the prevailing defence in most banking and payment-service-provider deployments, encode static threshold conditions that are brittle under adversarial adaptation. An attacker who can infer a threshold through controlled probing simply fragments transactions to remain sub-threshold. Legitimate users whose behaviour deviates temporarily — travel, device upgrade, festival spending — trigger false positives that degrade experience and erode platform trust. Machine learning directly addresses both failure modes: discriminative patterns are learnt from historical data rather than manually specified, complex non-linear feature interactions are captured automatically, and periodic retraining restores calibration as fraud patterns drift [4][5].

Despite progress, three challenges impede production deployment of ML-based UPI fraud detection. First, extreme class imbalance (fraud rates below 5% in well-defended systems) biases naive classifiers toward blanket approval. Second, the real-time latency requirement — scores must be returned in tens of milliseconds — limits model complexity and necessitates efficient inference engineering. Third, financial regulators increasingly mandate explainable automated decisions; a model that blocks a transaction must supply a human-readable rationale that a fraud analyst can validate or override [6][11].

This paper contributes a system that addresses all three challenges within a single integrated architecture: a dual-branch ensemble with calibrated fusion, a streaming inference pipeline meeting the 30 ms SLA, and SHAP/LIME explanations embedded directly in the real-time scoring path.

A. Paper Organisation

Looking back at earlier studies comes next in Section II, while pointing out what they missed guides why this one steps in. What stays unclear elsewhere becomes clear here through contrast.. Section III describes the proposed system architecture. Section IV details the data generation, preprocessing, and model training methodology. Section V presents experimental results. What comes next shows what was learned, where it fell short, then points ahead. Not every path worked out like first thought - some trails just ended. Ahead lies room to grow, though not everything fits yet into clear answers. Section VII concludes.

II. LITERATURE REVIEW AND RESEARCH GAPS

Negi et al. [1] surveyed ML techniques for UPI and telecom fraud and found that gradient-boosted methods — XGBoost and LightGBM — consistently delivered the highest accuracy on tabular payment datasets, with reported figures between 95% and 98%. The survey noted that ensemble configurations outperform single classifiers but rarely addressed real-time deployment constraints.

Bodade and Pawade [4] compared Logistic Regression (82%), Decision Tree (85%), and Random Forest (91%) on a synthetic UPI corpus, establishing useful baselines without exploring gradient-boosted or sequential architectures. Jeyachandran et al. [5] demonstrated that pipeline separation of feature computation and model inference reduces per-transaction latency to under 50 ms, influencing the streaming architecture in the present work.

Dave and Chudasama [13] compared Random Forest ($F1 \approx 0.89$), XGBoost ($F1 \approx 0.91$), and a stacking ensemble ($F1 \approx 0.93$), providing direct evidence that multi-model fusion yields measurable gains. Jagadeesan et al. [16] showed that LSTM models achieve higher recall than static classifiers on fraud types that manifest as behavioural sequences, supporting the inclusion of a sequential branch. Sharma et al. [11] integrated basic LIME explanations post-hoc, but the explainability component was applied to sampled predictions rather than embedded in the live scoring path.

Looking at past research reveals five repeated shortcomings. Public data built just for UPI is hard to find - most work leans on credit card info instead. Class imbalance often gets ignored or handled unevenly across papers. Metrics that matter in real systems, like speed and volume processed, rarely show up. Models combining table-style inputs with time-based patterns stay mostly unexplored. Explanations tend to come after decisions, tacked on loosely, rather than woven into how models operate. The present work directly addresses all five gaps.

III. PROPOSED SYSTEM ARCHITECTURE

The system follows a four-tier layered design that cleanly separates concerns while meeting the real-time latency constraint.

A. Tier 1 — Data Ingestion

During training, the ingestion tier reads the synthetic transaction corpus from a structured CSV file. During deployment, it accepts JSON payloads received at the Flask REST API endpoint, which simulates a Kafka-style message queue with a configurable buffer. All ingested events are immediately forwarded to the feature engineering tier without blocking the network handler.

B. Tier 2 — Feature Engineering

Raw transaction fields are transformed into eleven engineered features: transfer amount, transaction hour, day of week, device-change indicator, geographic-shift indicator, new-beneficiary indicator, transaction frequency in the preceding hour, time elapsed since last transaction, z-score deviation of current amount from the user's historical mean, collect-request flag, and payment-channel code. Continuous features are standardised with z-score normalisation using parameters fit exclusively on the training fold. A twist happens in training when a special SMOTE tool boosts the smaller group, lifting it from just 5% up to nearly 39%. This shift unfolds by crafting new examples - each born from blending a rare case with its five closest matches across measured traits.

C. Tier 3 — Dual-Branch Inference

The gradient-boosting branch (XGBoost-equivalent) receives the eleven-dimensional static feature vector and outputs a calibrated probability p_{xgb} . The model employs 200 sequential decision trees with maximum depth 5 and learning rate 0.1, trained with early stopping against a 10% validation holdout. The LSTM branch receives a (5, 8) sequential tensor constructed from the eight most temporally sensitive features across the five most recent transactions for the same user, and outputs a recurrent probability p_{lstm} . A single homemade LSTM built with NumPy runs on 24 internal nodes, learning across 30 full passes through the data while guided by a binary loss signal. Though coded from scratch, it sticks close to classic training patterns without extra frills or modern tweaks hiding underneath its structure.

One number comes from mixing two guesses using a balance knob set at 0.6, picked by testing every tenth step between zero and one. That mixed result gets checked against two lines you can move. Below 0.4? The system says yes without asking. Above 0.7? It stops the deal right there. In between, a person must take a look before any decision.

D. Tier 4 - Explanation and Routing

Every escalated or blocked transaction triggers the explainability module. A permutation-based SHAP approximation computes global feature importance across the portfolio. A LIME surrogate fits a locally faithful linear model around the individual prediction, identifying the top contributing factors in terms the analyst recognises. Both explanation types are appended to the JSON API response and written to a persistent, append-only audit log. Total explanation overhead is under 3 ms per event.

IV. METHODOLOGY

A. Synthetic Dataset Construction

A Python data-generator module creates 20,000 UPI transactions for 2,000 virtual users, each assigned a stable behavioural profile characterised by a habitual spending mean drawn from a log-normal distribution, a primary device identifier, a home city code, and a diurnal activity pattern. Legitimate transactions (95%, $n \approx 19,000$) are sampled from these profiles with controlled Gaussian noise. Fraudulent transactions (5%, $n \approx 1,000$) are distributed equally across five attack families and inject characteristic distortion patterns: QR-reversal events exhibit anomalous collect-request flags; SIM-swap events show sudden device-change and geographic-shift combinations; collect-abuse events carry elevated hourly frequency; phishing events present amount z-scores exceeding three standard deviations; account-takeover events simultaneously trip device-change, geographic-shift, new-beneficiary, and collect-request indicators.

B. Training Protocol

The Splitting the data happens using an 80-20 divide, keeping things balanced so the smaller group stays visible. That tiny slice - just five percent - is fraud cases, mirrored exactly in the testing portion. Metrics calculated later actually match real-world conditions because of this setup. Realism matters when measuring performance. The way it breaks down avoids distorting results by accident. SMOTE is applied exclusively within the training fold. Four baseline classifiers are trained and evaluated first: Logistic Regression (LR), Decision Tree (DT), Isolation Forest (IF), and Random Forest (RF). The gradient-boosting branch and LSTM branch are then trained individually, followed by grid-search optimisation of the fusion weight α . Every test runs with the same starting point by design. A set seed makes outcomes repeatable across trials. Reproducibility stays ensured through consistent initialization. Identical conditions support result stability throughout. The setup locks randomness at first step always.

C. Evaluation Metrics

What matters most in sorting results? Precision, recall, F1-score, along with ROC-AUC - these come straight from the uneven test batch. Performance during real use shows up in response times: half stay below P50, nearly all under P95, while steady QPS counts how many questions slip through each second. A thousand fake calls run at once, timed down to the billionth of a second using Python's clock. Each number reflects actual strain, nothing smoothed out.

V. EXPERIMENTAL RESULTS

A. Classification Performance

Looking at Table I, you can see how each model did on the test set using precision, recall, F1-score, and ROC-AUC. With the exception of Isolation Forest, every baseline hits an F1-score past 0.95 once SMOTE adjusts the balance - proof the synthetic data holds signal. Since its anomaly scores aren't aligned with the skewed test labels, Isolation Forest struggles badly when used in a supervised setup, landing at just 0.141 in F1. At the top, two stand out: Random Forest reaches 0.971, while GradientBoosting edges slightly ahead at 0.973. Midway through testing, the LSTM by itself hits an F1 score of 0.850 - clearly showing it struggles to cover every type of attack on its own. Instead of relying solely on sequence patterns, combining methods brings better balance; here, a mix of GradientBoosting's sharp accuracy on structured data and LSTM's strength in tracking sequences lifts performance sharply, landing at F1 = 0.958 and ROC-AUC nearly perfect at 0.9985.

Model	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	0.925	0.990	0.957	0.9998
Decision Tree	0.933	0.975	0.954	0.9861
Isolation Forest	0.170	0.120	0.141	0.9034
Random Forest	0.952	0.990	0.971	0.9999
GradientBoosting	0.957	0.990	0.973	0.9999
LSTM (Standalone)	0.820	0.880	0.850	0.9100
Hybrid ($\alpha = 0.6$)	0.938	0.979	0.958	0.9985

TABLE I. Model Performance Comparison

B. Operational Performance

Operational Metric	Measured Value
Dataset size	20,000 synthetic UPI transactions
Median latency (P50)	~12 ms
95th-percentile latency (P95)	~25 ms
Sustained throughput	~1,200 QPS
Explanation overhead	< 3 ms per flagged event
Top SHAP feature	new_beneficiary (importance = 0.38)

TABLE II. Operational Performance Metrics

Table II summarises the measured deployment metrics. Most queries finish in under 25 milliseconds, clearing the 30 ms mark needed for instant payment checks. Running steady at 1,200 queries per second happens when tested with 1,000 users active.

at once. SHAP and LIME explanation computation adds fewer than 3 ms per flagged transaction, keeping total latency well within budget. The globally most important feature is `new_beneficiary` (normalised SHAP importance 0.38), followed by `transaction_hour` and `amount_zscore`, consistent with domain-expert intuition about high-risk UPI scenarios.

C. SHAP Feature Importance

Permutation-based SHAP analysis identifies `new_beneficiary` (0.38), `transaction_hour` (0.21), `amount_zscore` (0.17), `collect_request` (0.13), and `device_changed` (0.11) as the five dominant predictors. This ranking aligns with findings in prior literature [9][22] and provides actionable interpretive guidance for fraud analysts reviewing escalated transactions. LIME local explanations for individual blocked transactions consistently highlight the same top features, validating cross-method consistency.

VI.DISCUSSION

A. Strengths

The hybrid architecture captures complementary evidence channels that no single model type covers: the gradient-boosting branch excels at tabular feature interactions such as amount deviation combined with device change, while the LSTM branch detects temporally structured patterns such as rapid-fire small transactions preceding a large account-takeover transfer. The fusion weight $\alpha = 0.6$ appropriately down-weights the LSTM branch, which has higher variance on the small synthetic corpus, while still incorporating its sequential evidence.

The integration of explainability directly into the real-time scoring path — rather than as an offline post-hoc analysis — is a key distinguishing characteristic of this work relative to the surveyed literature [11]. Every analyst-reviewed or auto-blocked transaction carries a complete explanation, enabling both regulatory compliance and closed-loop feedback collection for future retraining.

B. Limitations

All results are obtained on a controlled synthetic dataset. The distortion patterns injected for each fraud family are derived from published attack taxonomies [3] but cannot fully reproduce the adversarial complexity, label noise, or covariate shift present in live banking data. Generalisation to real operational data will require validation with banking partners. The custom LSTM is deliberately lightweight to satisfy the latency constraint; a transformer-based sequence model might achieve higher recall on long-range temporal dependencies at the cost of increased inference time.

C. Future Directions

Five extensions are identified for future work: (i) validation on real operational UPI data from banking partners; (ii) Graph Neural Networks for detecting mule-account rings by representing the payment network as a heterogeneous graph; (iii) federated learning to enable multi-institutional model training without centralising sensitive transaction records; (iv) transformer-based sequence models with self-attention to improve temporal branch recall; and (v) integration of multimodal evidence — customer complaint text, device telemetry, network metadata — to enrich the feature space against sophisticated multi-stage attacks.

VII.CONCLUSION

This paper presented a real-time UPI fraud detection system built on a hybrid ensemble that fuses gradient-boosted decision trees with a custom LSTM network through a calibrated weighted average. On a 20,000-transaction synthetic corpus spanning five UPI-specific attack families, the hybrid model achieved an F1-score of 0.958 and ROC-AUC of 0.9985, meeting all classification objectives. The deployment infrastructure — a Flask-based HTTPS REST API with bearer token authentication — delivers scoring at a 95th-percentile latency of 25 ms and 1,200 QPS throughput. SHAP global and LIME local explainability modules are embedded directly in the scoring path, adding under 3 ms overhead per flagged event and producing analyst-readable justifications for every contested decision. The adaptive retraining pipeline with statistical drift detection ensures sustained effectiveness as fraud patterns evolve. Together, these contributions advance the state of the art for deployable, transparent, and operationally efficient UPI fraud screening systems.

Acknowledgement

The authors gratefully acknowledge the guidance and support of Dr. Manjula R. Chougala, Assistant Professor, Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, Bangalore. The authors also thank the Department and the Institute for providing the computational infrastructure used in this research.

REFERENCES

- [1] A. Negi, D. Negi et al., "A Comprehensive Survey on Machine Learning Techniques for UPI and Telecommunication Fraud Detection," *International Journal of Engineering Research and Technology*, vol. 14, no. 11, 2025.
- [2] D. J. Lakshmi, "AI-Based UPI Scam Detection Using Crowdsourced Patterns," *International Journal of Innovative Research in Technology*, vol. 12, no. 3, 2025.
- [3] C. N. Bhavani and S. S. Saheb, "Fraudulent Threats and Mitigating Strategies in UPI Transactions," *Proceedings of the 12th International Conference on Emerging Trends in Corporate Finance and Financial Markets*, 2024.
- [4] S. S. Bodade and P. P. Pawade, "UPI Fraud Detection Using Machine Learning," *Journal of Emerging Technologies and Innovative Research*, vol. 11, no. 4, 2024.
- [5] P. Jeyachandran, A. S. V. Akisetty et al., "Real-Time Fraud Detection in Digital Payments Using Machine Learning," *Integrated Journal for Research in Arts and Humanities*, vol. 4, no. 6, 2024.

- [6] V. S. Mangali, P. Bhuvaneshwari et al., "Enhanced Detection of UPI Frauds Using Advanced Machine Learning Techniques," European Alliance for Innovation Proceedings, 2025.
- [7] D. J. Kumari, G. Tejaswi et al., "AI-Powered UPI Fraud Detection," International Journal of Innovative Science and Research Technology, vol. 10, no. 4, 2025.
- [8] K. R. Kavya and R. Usha Sree, "UPI Fraud Detection Using Machine Learning," International Journal of Advanced Research in Science, Communication and Technology, vol. 4, no. 6, 2024.
- [9] K. Padmanaban and S. Reshma, "UPI Fraud Detection Using Machine Learning," International Journal of Research and Analytical Reviews, vol. 12, no. 2, 2025.
- [10] K. Koli, I. Uge et al., "An Analytical Approach to UPI Fraud Detection Using Machine Learning and Deep Learning," International Journal of Innovative Research in Technology, vol. 11, no. 5, 2024.
- [11] H. Sharma, K. Sharma and R. Kumar, "UPI Fraud Detection System with LIME-Based Explainability," International Journal of Innovative Science and Research Technology, vol. 10, no. 6, 2025.
- [12] K. Koli, I. Uge et al., "Analytical Approaches to UPI Fraud Detection: An Updated Survey," International Research Journal of Modernization in Engineering Technology and Science, vol. 7, no. 4, 2025.
- [13] V. Dave and D. Chudasama, "Fraud Detection in UPI Transactions Using Ensemble Learning," International Journal of Creative Research Thoughts, vol. 13, no. 4, 2025.
- [14] S. Malode, A. Wankhede et al., "UPI Fraud Detection Using Machine Learning," International Journal for Research Publication and Seminar, vol. 16, no. 1, 2025.
- [15] S. S. Bodade and P. P. Pawade, "Review on UPI Fraud Detection Using Machine Learning," International Journal for Research in Applied Science and Engineering Technology, 2023.
- [16] S. Jagadeesan, K. S. Arjun et al., "UPI Fraud Detection Using LSTM-Based Sequence Modelling," ResearchGate Preprint, 2025.
- [17] S. R. M. Mehrooz et al., "Deployment Architectures for Real-Time UPI Fraud Scoring," International Journal of Advances in Engineering and Management, vol. 6, no. 3, 2024.
- [18] B. Sethi, S. Mhatre et al., "Machine Learning-Based UPI Fraud Detection Using Random Forest," Atlantis Press Proceedings, 2025.
- [19] Stripe, Inc., "How Machine Learning Works for Payment Fraud Detection and Prevention," Stripe Engineering Blog, 2024. [Online]. Available: <https://stripe.com/blog>
- [20] NPCI, "UPI Product Statistics," National Payments Corporation of India, 2024. [Online]. Available: <https://www.npci.org.in/what>