INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

Sentinel Pay: A Real-Time Machine Learning Model

Lakshya Sharma[1], Shivam Kumar[2], Gaurav Kumar[3], Ronak Singh[4], Mohammad vakil[5]

[1234]*Department of Computer Science (Data Science)*

[5]*Professor, Department of computer science*

RD Engineering College, Ghaziabad, Uttar Pradesh, India

ABSTRACT

The rapid digitisation of financial transactions in India has created fertile ground for sophisticated payment fraud. This paper presents Sentinel Pay, an ensemble-based real-time fraud detection framework capable of evaluating transactions across five payment modalities — Unified Payments Interface (UPI), credit cards, debit cards, net banking, and mobile wallets. We combine three complementary machine learning algorithms, namely Gradient Boosting, Random Forest, and Logistic Regression, through a soft-voting ensemble that assigns differential weights to each learner. To address class imbalance inherent in fraud datasets, we applied oversampling with resampling techniques during training. The resulting model analyses nineteen behavioural and contextual transaction features, generating a fraud risk score in under two milliseconds per query. In empirical evaluation on a held-out test set of 10,000 transactions, the system attained 99.5% accuracy, 99% precision, and 100% recall against fraud cases. Alongside the predictive model, we engineered a Flask-based monitoring dashboard with live transaction feeds, risk visualisations, and a standalone payment gateway simulator that demonstrates seamless REST API integration with external platforms.

Keywords: Payment fraud detection, ensemble learning, UPI security, real-time inference, SMOTE, Flask, machine learning.

1. INTRODUCTION

India's digital payment ecosystem has undergone a transformational shift over the past decade. The National Payments Corporation of India reported that UPI alone processed over ten billion transactions in a single month during 2024, a volume that would have been inconceivable at the technology's inception. While this democratisation of financial access is commendable, it simultaneously creates an expanded attack surface for malicious actors. Fraud incidents involving digital payments have grown commensurately, inflicting substantial monetary losses on consumers, merchants, and financial institutions alike.

Conventional rule-based fraud detection mechanisms, which flag transactions based on static thresholds, are increasingly inadequate against adversaries who probe and reverse-engineer these boundaries. The fundamental limitation of fixed rules is their inability to generalise: a rule calibrated for yesterday's fraud pattern is often obsolete by tomorrow. Machine learning offers a fundamentally different paradigm — rather than encoding human intuitions into explicit conditions, models discover discriminative patterns from historical data and can generalise these patterns to unseen examples.

This paper describes our design and implementation of Sentinel Pay; a production-oriented fraud detection system tailored to the nuances of the Indian digital payment landscape. Unlike much of the existing literature, which concentrates exclusively on credit card fraud in Western markets, our framework addresses the full spectrum of payment modalities prevalent in India and incorporates UPI-specific behavioural signals that have no analogue in card-centric systems.

1.1 Problem Statement

Fraud detection is fundamentally an imbalanced binary classification problem. Fraudulent transactions constitute a small minority of overall activity, often below five percent in practice. This imbalance has two adverse consequences: naive classifiers achieve high nominal accuracy simply by predicting the majority class, and learning algorithms may fail to allocate sufficient attention to the minority class during training. Additionally, the cost asymmetry between false negatives (missed fraud) and false positives (blocked legitimate transactions) is severe: missing fraud causes direct financial loss, while over-blocking erodes customer trust and imposes operational costs.

Three structural deficiencies characterise existing deployed systems. First, excessive false positive rates inconvenience legitimate customers, particularly during travel or atypical purchasing behaviour. Second, models are typically static, failing to adapt as fraud tactics evolve. Third, latency constraints are frequently ignored in academic work, despite real-time payment environments demanding decisions within milliseconds.

1.2 Research Objectives

Our work pursues the following concrete objectives:

- Develop a multi-modal fraud detection model achieving at least 95% accuracy across all payment types
- Maintain false positive rate below 5% to preserve a positive customer experience
- Deliver predictions within three milliseconds to meet real-time payment processing requirements
- Engineer a comprehensive web dashboard for operational fraud monitoring
- Demonstrate integration-readiness through a standalone payment gateway simulator

2. RELATED WORK

Research on automated fraud detection spans more than two decades, evolving from simple threshold-based heuristics to sophisticated deep learning architectures. We review the most relevant threads below.

2.1 Evolution of Fraud Detection Methods

Early fraud detection systems operated on hand-crafted rules derived from domain expert knowledge. While interpretable, these systems are brittle and require continuous manual maintenance as fraud patterns shift. The introduction of logistic regression and decision trees in the early 2000s marked a transition toward data-driven approaches, with reported accuracy improvements to the 85–90% range. A pivotal advancement came with ensemble methods. Random Forests and Gradient Boosting demonstrated that combining multiple weak learners into a single strong classifier yields substantial accuracy gains while also reducing variance. These techniques became the dominant paradigm in applied fraud detection throughout the 2010s.

Chawla et al. (2002) identified class imbalance as a root cause of poor fraud recall and proposed

SMOTE, which generates synthetic minority-class examples by interpolating between existing samples in feature space. Their technique remains widely adopted, though later work by Lemaitre et al. (2017) demonstrated that combined over- and under-sampling strategies can further improve decision boundary quality.

2.2 Gaps in Existing Literature

Despite a mature body of work, several important dimensions remain underexplored:

- The vast majority of published studies focus exclusively on credit card fraud, leaving UPI and wallet-based fraud detection largely unaddressed
- Multi-modal systems capable of handling diverse payment instruments within a unified scoring framework are absent from the literature
- Real-time deployment considerations — latency budgets, API design, dashboard tooling — are rarely discussed alongside modelling results
- RBI-specific regulatory context and India-centric fraud patterns (e.g., new UPI payee exploitation) are not represented in datasets or feature sets

Sentinel Pay is designed to address precisely these gaps.

3.SYSTEM DESIGN AND IMPLEMENTATION

3.1 Architectural Overview

Sentinel Pay comprises four loosely coupled components that communicate via well-defined interfaces:

1. Synthetic Data Generator — Produces realistic transaction records for both training and simulation, parameterised by payment type and fraud probability
2. Machine Learning Engine — An ensemble model that ingests transaction feature vectors and returns calibrated fraud probability scores
3. Flask Web Dashboard — A browser-accessible interface providing live transaction monitoring, risk visualisation, and manual testing capabilities
4. Payment Gateway Simulator — A standalone Flask application on port 5001 that emulates external payment platform integration via REST API calls

End-to-end transaction evaluation follows a synchronous pipeline: a transaction arrives, features are extracted and normalised, the ensemble model scores the transaction, and the result is routed back to the calling system — all within two milliseconds under typical load.

3.2 Dataset Construction

As access to real transaction data was precluded by privacy constraints, we constructed a synthetic dataset of 50,000 transactions calibrated to match empirically observed fraud characteristics drawn from Reserve Bank of India annual reports and published academic datasets. The fraud rate was set at five percent, consistent with industry estimates for digital payment fraud in India.

Legitimate transaction generation modelled realistic behavioural patterns: amounts were drawn from log-normal distributions reflecting each payment type's typical range, timestamps were concentrated in daytime hours, and device and location attributes were kept consistent with established customer profiles. Fraudulent transactions were generated with deliberately distinct characteristics — anomalously high or suspiciously low amounts, activity concentrated in overnight hours, elevated transaction velocity, use of VPN connections, and preference for high-

risk merchant categories such as cryptocurrency exchanges and online casinos.

Critically, feature distributions for fraudulent and legitimate transactions were designed with overlapping ranges to prevent any single feature from acting as a perfect separator. This ensures the model must learn genuinely multivariate decision boundaries rather than exploiting degenerate single-feature shortcuts.

3.3 Feature Engineering

Feature selection was informed by a combination of domain knowledge, RBI fraud advisory literature, and iterative experimentation. The final feature set of nineteen variables spans five conceptual categories:

Basic Transaction Attributes: payment modality, merchant category, transaction amount, and a binary high-risk merchant indicator covering crypto exchanges, wire transfer services, and gift card platforms.

Temporal Features: hour of day, day of week, weekend indicator, and an odd-hour flag that activates for transactions occurring between 10 PM and 6 AM, a period associated with elevated fraud incidence.

Geographic and Device Signals: international transaction flag, new device fingerprint indicator, and VPN/proxy usage detection.

Behavioural Features: transaction velocity (count of transactions in the preceding hour), new UPI payee indicator, failed authentication attempt count, and first-time payment to recipient flag.

Statistical Features: the ratio of transaction amount to the customer's historical average spend (amount-to-average ratio), and the customer's baseline average transaction amount. The amount-to-average ratio proved particularly discriminative: a transaction of ₹45,000 from a customer whose historical average is ₹600 carries qualitatively different risk than the same amount from a high-frequency corporate account.

3.4 Model Training and Ensemble Architecture

We evaluated three base learners independently prior to ensemble construction:

- Logistic Regression: interpretable baseline, achieved 92.3% accuracy in isolation
- Random Forest (100 estimators, max depth 8): achieved 96.8% accuracy with good generalisation
- Gradient Boosting (150 estimators, learning rate 0.08): achieved 97.9% accuracy with strong discrimination on complex non-linear patterns

The three learners were combined into a soft-voting ensemble with weights of 3, 2, and 1 for Gradient Boosting, Random Forest, and Logistic Regression respectively, reflecting their relative empirical performance. Soft voting aggregates

predicted class probabilities rather than hard class labels, which yields better-calibrated confidence scores — a property important for the intermediate risk tiers (review and caution) that fraud analysts must manually triage.

To address the five-to-one class imbalance in training data, we applied oversampling of the minority class using random resampling with replacement, creating a balanced 50:50 training distribution. This technique was applied exclusively to the training partition; the test set retained its natural class distribution to yield unbiased evaluation metrics. A standard 80:20 train-test split with stratification preserved fraud prevalence across both partitions.

3.5 Risk Scoring and Decision Thresholds

Model output probabilities are multiplied by 100 to produce a fraud risk score on a 0–100 scale, which is then mapped to four operational tiers:

Risk Score Range	Risk Level	Decision	Recommended Action
0 – 34	Low	Approved	Process normally
35 – 49	Medium	Review	Log for analyst review
50 – 69	High	Flagged	Request additional verification
70 – 100	Critical	Blocked	Decline and alert customer

Table 1: Fraud Risk Scoring Tiers and Operational Responses

3.6 Dashboard and Payment Gateway

The operational dashboard was implemented in Flask with a dark-themed interface that prioritises readability under sustained analyst use. A background simulation thread generates one new transaction every 1.5 seconds, populating a live feed that displays risk scores, customer details, and colour-coded status indicators. An administrative panel provides model performance metrics and supports on-demand model retraining without service interruption.

The payment gateway simulator runs as an independent process on port 5001, accepting payment form submissions and forwarding them to the fraud detection API at /Api/fraud-check. Its JSON contract mirrors the format used by commercial Indian payment aggregators such as Razor pay and PayU, demonstrating that Sentinel Pay can be integrated into existing platforms with minimal adapter code. The gateway interface exposes all nineteen model input features through an interactive form, enabling comprehensive manual testing of diverse fraud scenarios.

4. EXPERIMENTAL RESULTS

4.1 Classification Performance

Table 2 summarises the performance of each component algorithm and the final ensemble on the held-out 10,000-transaction test set.

Model	Accuracy (%)	Precision (%)	Recall (%)
Logistic Regression	92.3	88.1	84.6
Random Forest	96.8	94.2	93.7
Gradient Boosting	97.9	96.5	95.8
Sentinel Pay Ensemble	99.5	99.0	100.0

The ensemble correctly identified all 500 fraudulent transactions in the test set (zero false negatives) while generating only 50 false positives among the 9,500 legitimate transactions, yielding a false positive rate of 0.53%. This performance implies that in a hypothetical deployment processing 100,000 daily transactions at a five percent fraud rate, the system would prevent all 5,000 fraud attempts while incorrectly blocking only 53 legitimate transactions — an operationally acceptable trade-off.

4.2 Feature Importance Analysis

Post-training feature importance analysis across the Gradient Boosting and Random Forest components revealed the following ranking:

- Amount-to-average ratio (28%): The ratio of current transaction amount to the customer's historical average proved the strongest single discriminator, capturing the anomalous spending magnitude that characterises many fraud scenarios
- Transaction velocity (22%): High transaction frequency within a short

window strongly indicates card testing or account takeover attempts

- High-risk merchant category (18%): Purchases at cryptocurrency exchanges, wire transfer platforms, and online casinos carry disproportionately elevated fraud probability
- New device fingerprint (12%): First-time device usage correlates with account takeover scenarios where fraudsters use their own hardware
- International transaction flag (8%): Cross-border transactions, particularly in combination with other risk signals, require heightened scrutiny

The temporal features — hour of day and odd-hour flag — contributed meaningfully but were not dominant in isolation, confirming the model correctly weights combinations of signals over individual indicators.

4.3 Inference Latency

Across 1,000 repeated measurements on a consumer-grade laptop, mean prediction latency was 1.7 milliseconds with a 99th percentile of 2.4 milliseconds. This comfortably satisfies the three-millisecond budget specified in our objectives and is sufficient to support approximately 500 queries per second on a single machine without batching optimisations.

5. IMPLEMENTATION CHALLENGES

5.1 Python Version Compatibility

Our development environment ran Python 3.13, which introduced breaking changes in several dependency libraries. NumPy 2.0, in particular, altered the behaviour of probability distribution functions in ways that caused silent numerical errors in our initial data generator. Resolving this required careful library version pinning and partial rewrites of sampling logic to use distribution APIs compatible across NumPy versions.

5.2 Class Imbalance Mitigation

Our initial training runs produced a model with 90% nominal accuracy that simply predicted 'legitimate' for every transaction — exploiting the natural class distribution rather than learning discriminative patterns. We diagnosed this through per-class recall metrics rather than aggregate accuracy, then applied oversampling restricted strictly to the training partition. Applying any resampling technique to the test set would introduce evaluation bias and inflate reported metrics.

5.3 Avoiding Degenerate Feature Separation

An early version of the synthetic data generator assigned non-overlapping ranges to the mins_since_last_txn feature for fraudulent and legitimate transactions. This inadvertently created a single-feature separator: the model learned to classify transactions entirely by this one variable, ignoring all other features. We identified this through feature importance inspection and corrected it by ensuring realistic distributional overlap across all features, forcing the model to utilise the full feature space.

5.4 Real-Time Dashboard Architecture

Our initial design used WebSocket connections for live updates, but we encountered persistent compatibility issues between Flask-Socket IO and the async event loop under Python 3.13. We pivoted to a polling architecture in which the client fetches updates from the server every two seconds. This approach proved more robust and introduced negligible additional latency for operational purposes.

6. LIMITATIONS AND FUTURE DIRECTIONS

6.1 Current Limitations

- Synthetic training data: Despite calibration against published fraud statistics, synthetic data cannot fully replicate the distributional complexity of real transaction logs, which may contain fraud patterns We did not anticipate
- Static model: The trained model does not update incrementally as new transactions arrive. In adversarial environments where fraud patterns evolve rapidly, periodic retraining pipelines are essential
- Shallow behavioural history: Features are computed over short windows. Richer longitudinal profiles — spending patterns over months, location sequences, merchant graphs — could substantially improve discrimination
- No graph-based signals: Fraud rings frequently reuse devices, phone numbers, and beneficiary accounts across multiple victims. Graph neural networks operating over transaction networks could surface these structural patterns

6.2 Planned Extensions

- LSTM-based sequence modelling to capture temporal transaction patterns and detect behavioural drift
- Graph Neural Network integration for account-level fraud ring detection
- Automated weekly retraining pipeline with drift detection to maintain model currency
- SHAP-value explainability dashboard enabling analysts to interrogate individual prediction factors
- A/B testing infrastructure to evaluate model updates on a traffic subset before full deployment

7. CONCLUSION

This paper presented Sentinel Pay, a machine learning-based fraud detection framework designed for India's multi-modal digital payment environment. Through careful feature engineering, ensemble model construction, and deliberate attention to class imbalance, we achieved 99.5% accuracy and perfect fraud recall on a held-out evaluation set, while maintaining an inference latency below two milliseconds.

Several lessons from this work carry broader applicability. First, feature quality consistently outweighs model complexity: our nineteen carefully constructed features, particularly the amount-to-average ratio, delivered stronger discriminative power than a naively specified deep model would have on the same data. Second, evaluation methodology must account for class imbalance — per-class recall and precision metrics reveal model behaviour that aggregate accuracy conceals. Third, synthetic data generation requires deliberate distributional overlap between classes; inadvertent separation creates degenerate models that appear high-performing but have learned shortcuts rather than genuine patterns.

Sentinel Pay's architecture — decoupled components communicating over REST APIs, an operational dashboard, and a payment gateway simulator — reflects the engineering requirements of production deployment beyond academic benchmarking. We believe this orientation distinguishes the work from purely experimental studies and provides a foundation for practical adoption by financial institutions and payment service providers operating in the Indian market.

REFERENCES

[1] Reserve Bank of India. (2024). Annual Report on Payment Systems and Settlement Systems in India. Retrieved from https://rbi.org.in

[2] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.

[3] Bahnsen, A. C., Stojanovic, A., Aouada, D., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. Expert Systems with Applications, 51, 134–142.

[4] Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with under sampling for unbalanced classification. In IEEE Symposium on Computational Intelligence and Data Mining (pp. 159–166).

[5] Wang, C., & Xu, D. (2018). A hybrid ensemble method for credit card fraud detection. In Proceedings of the International Conference on Machine Learning and Cybernetics.

[6] Lemaitre, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research, 18(17), 1–5.

[7] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.

[8] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232.

[9] National Payments Corporation of India. (2024). UPI Product Statistics. Retrieved from https://www.npci.org.in/what-we-do/upi/upi-ecosystem-statistics