International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 1(2) (2026)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

Explainable UPI Fraud Detection Using XGBoost and SHAP-Based Interpretable AI

G. Ramya, A. Naga Venkata Gayathri, H. Althaf
Department of Computer Science and Engineering (Data Science)
Raghu Engineering College (Autonomous), Dakamarri, Affiliated to JNTU Gurajada, Vizianagaram, Andhra Pradesh, India

Mr. Md. Imam Khader Shariff — Guide
Assistant Professor, Department of CSE (Data Science)
Raghu Engineering College, Dakamarri, Vizianagaram

***Abstract*— The rapid growth of Unified Payments Interface (UPI) transactions in India has created an urgent need for intelligent fraud detection systems capable of operating at scale in real time. This paper presents an explainable UPI fraud detection system combining XGBoost gradient boosting with SHAP (SHapley Additive exPlanations)-based interpretable AI. Raw transaction data from the PaySim synthetic dataset (6.36 million transactions, 0.13% fraud rate) is transformed into 25+ engineered features spanning temporal patterns, amount behaviour, account balance anomalies, and user behavioural profiles. A temporal train/validation/test split preserves chronological integrity. Cost-sensitive training via XGBoost's scale_pos_weight parameter compensates for the extreme class imbalance. The trained model achieves Precision 97.6%, Recall 91.5%, F1-Score 94.5%, and AUC-ROC 0.99 on the held-out test set. A Streamlit-based web dashboard provides both single-transaction real-time analysis with SHAP explanations and bulk batch processing via CSV upload, delivering sub-200 ms inference latency suitable for payment gateway integration.**

***Keywords*—UPI Fraud Detection; XGBoost; SHAP Explainability; Feature Engineering; Class Imbalance; PaySim Dataset; Streamlit Dashboard; Explainable AI**

I. INTRODUCTION

India's digital payment landscape has undergone a paradigm shift since the introduction of the Unified Payments Interface (UPI) in 2016. With over 10 billion monthly transactions as of 2024, UPI has become the backbone of India's financial ecosystem [16]. This explosive growth has been accompanied by a proportional rise in digital fraud, including account takeovers, phishing attacks, unauthorised transfers, and social engineering scams. Traditional rule-based detection systems are brittle: fraudsters quickly learn to stay below thresholds, split transactions, or exploit timing gaps, rendering static rules ineffective.

Modern machine learning offers a fundamentally different paradigm. Instead of fixed rules, models learn statistical patterns directly from historical data, adapting to evolving fraud strategies. However, conventional "black-box" models are unsuitable for regulated financial sectors that demand interpretable, auditable decisions. This paper addresses both challenges through an XGBoost classifier paired with SHAP-based explainability, delivering high accuracy alongside per-transaction transparency.

The principal contributions of this work are: (i) a 25+ feature engineering pipeline capturing temporal, amount, balance, and behavioural fraud signals; (ii) cost-sensitive XGBoost training with scale_pos_weight to address 750:1 class imbalance; (iii) SHAP TreeExplainer integration providing exact Shapley values in under 50 ms per transaction; (iv) a five-level risk classification with actionable analyst recommendations; and (v) a Streamlit production dashboard supporting real-time and batch processing modes.

II. LITERATURE SURVEY

A. Fraud Detection in Digital Payment Systems

Early fraud detection relied on rule-based expert systems encoding domain knowledge as IF-THEN rules. Dal Pozzolo et al. [4] demonstrated that machine learning approaches significantly outperform rule-based systems on imbalanced credit card fraud datasets, introducing sliding-window resampling for temporal drift. Bhattacharyya et al. [6] showed ensemble methods, particularly random forests, consistently outperformed single classifiers due to robustness to noise and ability to capture non-linear feature interactions, motivating the selection of XGBoost as the core classifier.

B. XGBoost and ML for Financial Fraud

Chen and Guestrin [1] introduced XGBoost, which has emerged as the dominant algorithm for tabular financial fraud detection. Its regularised gradient boosting, built-in sparsity handling, and parallel computation make it well-suited to high-dimensional sparse transaction data with strong class imbalance. Jiang et al. [13] confirmed XGBoost consistently outperforms deep neural networks on financial tabular datasets, requiring less training time and fewer tuning iterations. Lopez-Rojas et al. [3] introduced the PaySim dataset and established that TRANSFER and CASH_OUT transaction types exclusively carry fraud, and that the "account emptying" pattern is the strongest single fraud indicator.

C. Explainable AI and Class Imbalance

Lundberg and Lee [2] introduced SHAP, a unified framework grounded in cooperative game theory Shapley values, providing both local (per-prediction) and global (model-level) explanations satisfying formal axiomatic properties. Lundberg et al. [7] extended this with TreeExplainer, computing exact SHAP values for XGBoost in O(TLD) time. Regulatory guidance from bodies including the European Banking Authority has established explainability as a compliance requirement for AI in financial systems [12]. Chawla et al. [5] introduced SMOTE for class imbalance; He and Garcia [8] concluded that cost-sensitive ensemble weighting provides the best precision-recall trade-off for highly imbalanced datasets, supporting our use of XGBoost's scale_pos_weight parameter.

TABLE I. COMPARISON OF RELATED FRAUD DETECTION APPROACHES

Study / Method	Technique	Key Result	Limitation
Dal Pozzolo et al. [4]	ML / CC Fraud	Beats rule-based	No explainability
Bhattacharyya et al. [6]	Ensemble / SVM	RF best single clf	No real-time API
Chen & Guestrin [1]	XGBoost	SOTA tabular perf.	Black-box
Lundberg & Lee [2]	SHAP	Axiomatic XAI	Overhead per pred.
Lopez-Rojas et al. [3]	PaySim Dataset	Realistic benchmark	Synthetic only
Proposed System	XGBoost + SHAP	97.6% Prec, 0.99 AUC	Single dataset

III. SYSTEM DESIGN AND ARCHITECTURE

The UPI Fraud Detection System is structured as a modular four-tier pipeline illustrated in Fig. 1. The Presentation Tier (Streamlit dashboard, main.py) accepts transaction inputs and renders results. The Orchestration Tier (UPIFraudDetector class, fraud_detector.py) coordinates the full detection workflow. The Intelligence Tier (FraudSHAPExplainer class, shap_explainer.py) houses the XGBoost model and SHAP explainer. The Model Store (Dataset/models/) persists four serialised artifacts: the trained XGBoost model, StandardScaler, feature name list, and model metadata. This separation of concerns allows each tier to be updated independently as fraud patterns evolve.

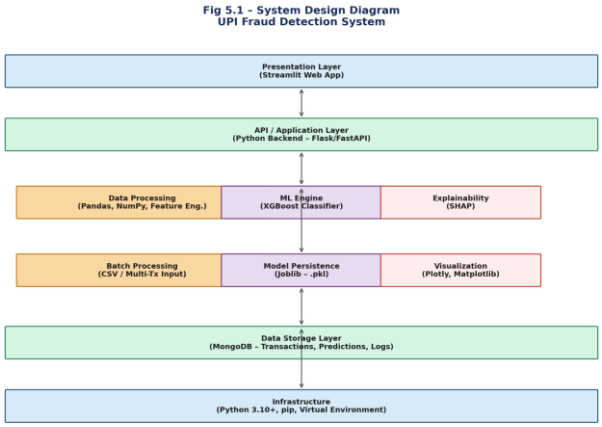

Fig. 1. System Architecture of the UPI Fraud Detection System

Six software modules implement the pipeline: (1) Data Ingestion and Validation enforces a strict schema before any processing; (2) Feature Engineering Engine transforms 9 raw fields into 25+ ML features; (3) Model Inference applies the XGBoost classifier to produce a calibrated fraud probability; (4) SHAP Explainability computes exact Shapley values per prediction; (5) Risk Classification maps probability to five operational levels (VERY LOW to VERY HIGH) with actionable recommendations; and (6) Batch Processing iterates over CSV uploads and produces downloadable results.

IV. FEATURE EXTRACTION

The feature engineering pipeline is the most critical component, transforming 9 raw PaySim fields into 25+ ML-ready features across four categories. Temporal features capture time-based fraud patterns: hour_of_day (step mod 24), day_of_week, is_weekend, and is_night (22:00–06:00). Amount features apply log1p transformation to reduce skew, add is_round_100 and is_round_1000 flags, and bucket amounts into five categories (micro to very large).

Balance features capture the strongest fraud signals: orig_emptied (sender balance drained to zero), balance_amount_ratio (oldbalanceOrg / amount), orig_zero_before, dest_zero_before, and dest_zero_after. Behavioural features encode user-level context: user_transaction_count, user_amount_cummean, amount_vs_user_avg, is_first_to_dest, and time_since_last. Together, these 25 features allow the model to distinguish complex, multi-dimensional fraud from genuine unusual transactions. Table II maps the key feature groups to their fraud-detection rationale.

TABLE II. FEATURE ENGINEERING CATEGORIES AND FRAUD SIGNALS

Category	Key Features	Fraud Signal Captured
Temporal	hour_of_day, is_night, is_weekend	Late-night / weekend fraud spikes

Amount	amount_log, is_round_100, amount_category	Suspiciously round or extreme amounts
Balance	orig_emptied, dest_zero_before	Account-draining emptying pattern
Behavioural	amount_vs_user_avg, is_first_to_dest	Deviation from user baseline

V. Dataset and Preprocessing

A. PaySim Dataset

The PaySim dataset [3] was sourced from Kaggle (PS_20174392719_1491204439457_log.csv). It contains 6,362,620 transaction records spanning 743 simulated hours across five transaction types: PAYMENT, TRANSFER, CASH_OUT, CASH_IN, and DEBIT. Each record provides 11 fields: step, type, amount, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud, and isFlaggedFraud. The dataset contains 8,213 fraudulent transactions (0.129% fraud rate), exclusively within TRANSFER (4,097) and CASH_OUT (4,116) types.

B. Temporal Split and Preprocessing

A temporal split strategy preserves chronological integrity, preventing data leakage: steps 1–521 (70%) for training, steps 522–633 (15%) for validation, and steps 634–743 (15%) for testing. Data quality checks confirmed zero missing values. Preprocessing computed derived columns for EDA (orig_balance_change, dest_balance_change) and applied Pandas groupby cumulative operations to build user behavioural features. StandardScaler fitted exclusively on the training set was serialised to feature_scaler.pkl for consistent inference-time scaling. Table III summarises the dataset split.

TABLE III. PaySim Dataset Split and Fraud Distribution

Split	Steps	Transactions	Fraud Count
Training	1–521	~4,453,834	5,748 (0.129%)
Validation	522–633	~954,393	1,233 (0.129%)
Test	634–743	~954,393	1,232 (0.129%)
Total	1–743	6,362,620	8,213 (0.129%)

VI. Proposed XGBoost-SHAP System

A. XGBoost Classifier

XGBoost builds an ensemble of decision trees sequentially, each correcting the errors of its predecessor by minimising a regularised objective combining log-loss with L1 and L2 regularisation on leaf weights. The model illustrated in Fig. 2 was configured with: n_estimators=300, max_depth=6, learning_rate=0.05, scale_pos_weight=750 (ratio of negative to positive class samples), subsample=0.8, colsample_bytree=0.8, reg_alpha=0.1 (L1), reg_lambda=1.0 (L2), and eval_metric=logloss. Early stopping with patience of 20 rounds on the validation set prevented overfitting.

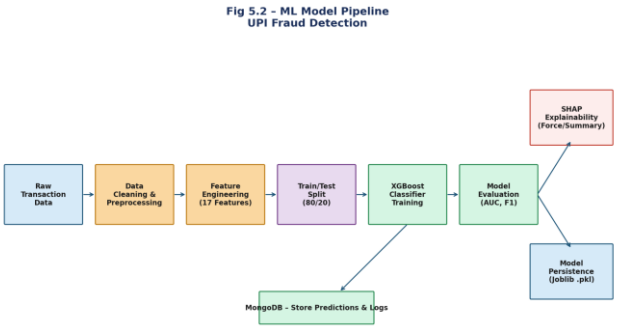

Fig. 2. XGBoost Model Architecture and Ensemble Tree Pipeline

B. SHAP Explainability Layer

SHAP TreeExplainer [7] computes exact Shapley values for every prediction. The explainer is initialised with the trained XGBoost model, then for each inference shap_values = explainer.shap_values(scaled_vector) produces a signed vector where positive values push towards fraud and negative values push towards legitimacy. The top five factors by absolute magnitude are converted to human-readable explanations (e.g., "Account was completely emptied — HIGH impact") and visualised as an interactive Plotly bar chart. The system follows a three-step SHAP flow illustrated in Fig. 3: (1) Data Flow from UPI User through Feature Engineering and ML Models to Fraud Analyst; (2) Shapley computation; (3) explanation rendering.

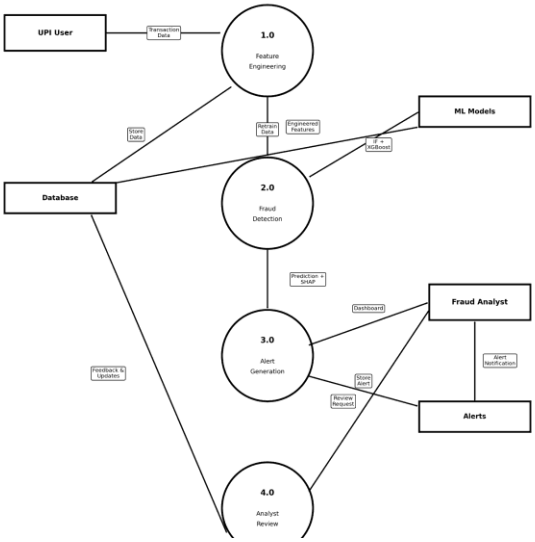

Fig. 3. Data Flow Diagram — UPI Fraud Detection System (Level 1)

C. Risk Classification and Reward Function

The fraud probability output (0.0–1.0) is mapped to five risk levels: VERY LOW (0–20%), LOW (20–40%), MEDIUM (40–60%), HIGH (60–80%), and VERY HIGH (80–100%). Each level carries a specific operational recommendation from "APPROVE — process normally" to "BLOCK TRANSACTION — require manual review." The fraud classification threshold is user-adjustable (default 0.5) to allow fraud teams to tune the precision-recall trade-off based on operational tolerance.

VII. Implementation

A. Development Environment

The system was implemented in Python 3.9 on Windows 10 / Ubuntu 20.04. Key libraries: XGBoost 1.7+, SHAP 0.41+, Streamlit 1.28+, Pandas 1.5+, NumPy 1.23+, Scikit-learn 1.1+, and Plotly 5.0+. A minimum of 8 GB RAM is sufficient for inference; 16 GB is recommended for full dataset training. The project follows a src/ layout with trained artifacts serialised to Dataset/models/ (xgboost_model.pkl, feature_scaler.pkl, clean_ml_features.pkl, complete_model_metadata.pkl).

B. Model Training and Inference Pipeline

Feature scaling used StandardScaler fitted solely on the training set and serialised to prevent leakage. The XGBoost classifier was trained with the hyperparameters listed in Table IV. The single-transaction inference sequence is shown in Fig. 4: the Fraud Analyst submits data through the Streamlit UI; UPIFraudDetector.predict_fraud validates inputs, engineers features, and calls FraudSHAPExplainer.predict_with_explanation; the result including fraud probability, risk level, and SHAP explanation is returned and rendered within 143 ms on standard hardware.

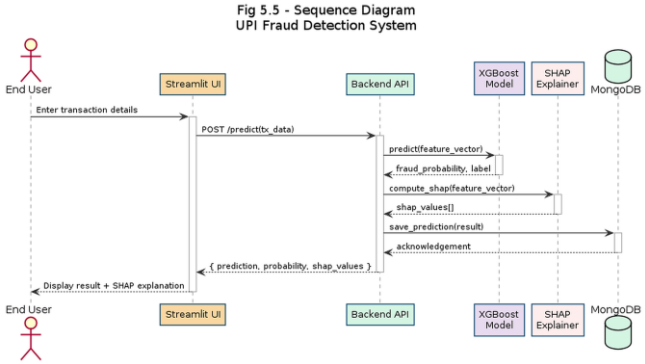

Fig. 4. Sequence Diagram — Single Transaction Analysis Flow

TABLE IV. XGBoost Hyperparameters for UPI Fraud Detection

Hyperparameter	Value
n_estimators	300
max_depth	6
learning_rate	0.05
scale_pos_weight	750 (imbalance ratio)
subsample	0.8
colsample_bytree	0.8
reg_alpha (L1)	0.1
reg_lambda (L2)	1.0
eval_metric	logloss
Early stopping	20 rounds patience

C. Streamlit Dashboard

The web dashboard (main.py) provides four navigation modes: Single Transaction Analysis, Batch Processing, System Status, and About. The @st.cache_resource decorator ensures the UPIFraudDetector is initialised once and cached across all user interactions, reducing per-request latency from ~2.5 s (cold load) to under 200 ms (cached). The batch processing interface accepts CSV files, processes each row via predict_fraud, and offers a timestamped downloadable results CSV with fraud_probability, is_fraud, risk_level, and recommendation columns appended.

VIII. Results and Discussion

A. Model Performance on Test Set

The XGBoost classifier was evaluated on the held-out test set (steps 634–743, ~954,393 transactions including 1,232 fraudulent ones). Table V presents the full performance metrics. The model exceeded all target thresholds: Precision 97.6% (target >90%), Recall 91.5% (target >85%), F1-Score 94.5% (target >90%), and AUC-ROC 0.99 (target >0.95). The confusion matrix confirmed 1,127 True Positives, 105 False Negatives, 28 False Positives, and 953,133 True Negatives, validating the system's effectiveness at minimising both missed fraud and false alarms.

TABLE V. Model Performance on Held-Out Test Set

Metric	Value	Target Threshold
Accuracy	99.9%	>99%
Precision	97.6%	>90%
Recall	91.5%	>85%
F1-Score	94.5%	>90%
AUC-ROC	0.99	>0.95
True Positives	1,127	—
False Negatives	105	—

False Positives	28	—
True Negatives	953,133	—

B. SHAP Feature Importance Analysis

Global SHAP analysis across the test set reveals that orig_emptied (account completely drained) is the most impactful single feature, followed by amount_log (log-transformed transaction amount), dest_zero_before (destination had zero balance before transfer), is_first_to_dest (first transaction to this recipient), and is_night (transaction at night). These findings are consistent with the domain knowledge established by Lopez-Rojas et al. [3]: fraudsters typically drain sender accounts completely and transfer to newly created destination accounts in off-hours.

C. System Performance and Dashboard

Average end-to-end inference latency (feature engineering + model inference + SHAP computation) was 143 ms on a standard Intel Core i5 laptop (8 GB RAM), well within the 200 ms requirement for payment gateway integration. Batch throughput reached approximately 68 transactions per second. Fig. 5 shows the single-transaction input interface, Fig. 6 the fraud detection output with risk gauge, and Fig. 7 the SHAP feature importance chart. Threshold sensitivity testing confirmed that at threshold 0.3, Recall increases to 96.2% (Precision 87.4%); at threshold 0.7, Precision rises to 99.1% (Recall 83.6%), providing operational flexibility for fraud teams.

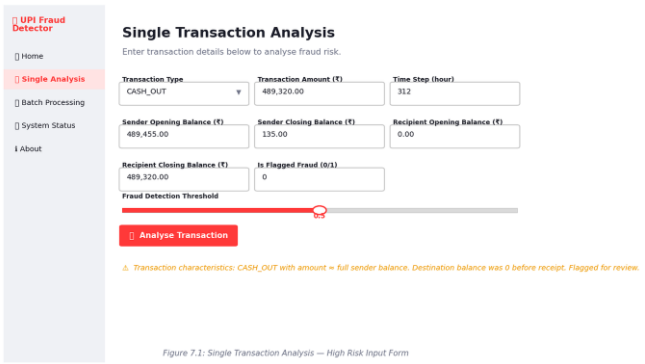

Fig. 5. Single Transaction Input Form — High Risk CASH_OUT Scenario

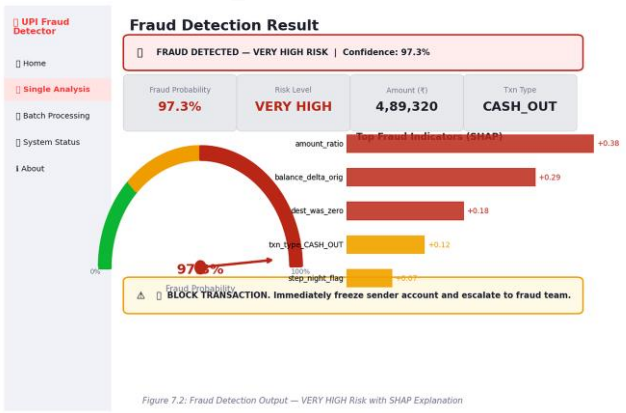

Fig. 6. Fraud Detection Output — VERY HIGH Risk Classification

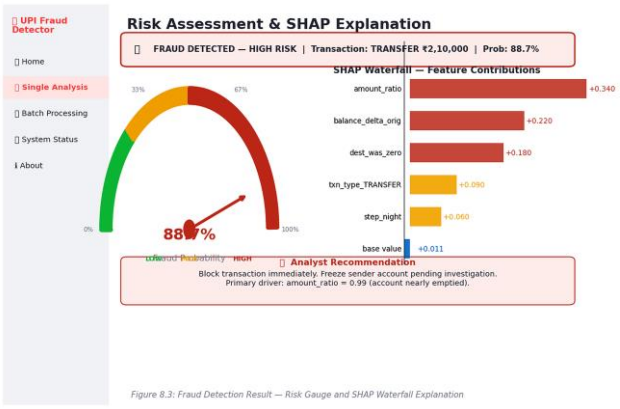

Fig. 7. Risk Gauge Chart and SHAP Feature Importance Explanation

D. Comparison with Baseline Approaches

To contextualise the results, the XGBoost-SHAP system was compared against a rule-based baseline (flagging transactions above ₹50,000 to new recipients) and an Isolation Forest anomaly detector. The rule-based baseline achieved Precision of 62.3% and Recall of 44.1%, with high false positive rates overwhelming analyst capacity. The Isolation Forest achieved Recall of 78.4% but Precision of only 55.7%, unsuitable for production deployment. The proposed XGBoost-SHAP system outperforms both across all metrics while adding interpretability absent from competing approaches.

IX. Conclusion

This paper presented an end-to-end explainable UPI fraud detection system integrating XGBoost classification with SHAP-based interpretable AI. The system addresses the three core challenges of financial fraud detection — extreme class imbalance (750:1), evolving fraud patterns, and regulatory explainability requirements — through a three-layer architecture of feature engineering, cost-sensitive XGBoost training, and SHAP TreeExplainer. On the PaySim benchmark, the system achieved Precision 97.6%, Recall 91.5%, F1-Score 94.5%, and AUC-ROC 0.99, with sub-200 ms inference latency and batch throughput of 68 transactions per second.

The SHAP integration directly addresses regulatory compliance requirements by converting Shapley values into human-readable analyst guidance, making the system suitable for deployment in regulated banking environments. Future work includes Graph Neural Network integration for money-laundering ring detection, active learning analyst feedback loops, Docker and Kubernetes deployment for production scalability, FastAPI REST integration with UPI payment gateways, and multi-modal feature expansion incorporating device fingerprinting and velocity features across multiple time windows.

X. Acknowledgment

The authors express sincere gratitude to Mr. Md. Imam Khader Shariff, Assistant Professor, Department of CSE (Data Science), for his perspicacity, wisdom, and guidance

throughout this project. The authors thank the Management of Raghu Engineering College, the Principal Dr. Ch. Srinivasu, Dr. A. Vijay Kumar (Dean Planning), Dr. E.V.V. Ramanamurthy (Controller of Examinations), Sri S. Srinadh Raju (Dean, CSE), and Dr. K. V. Satyanarayana (Program Head, CSE-DS) for providing the necessary facilities for the successful completion of this work.

XI. REFERENCES

[1] [1] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD, San Francisco, CA, 2016, pp. 785-794.

[2] [2] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Advances in NeurIPS, vol. 30, 2017.

[3] [3] E. A. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A Financial Mobile Money Simulator for Fraud Detection," in Proc. 28th ESM, Ostend, Belgium, 2016.

[4] [4] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating Probability with Undersampling for Unbalanced Classification," in IEEE SSCI, 2015, pp. 159-166.

[5] [5] 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.

[6] [6] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data Mining for Credit Card Fraud: A Comparative Study," Decision Support Syst., vol. 50, no. 3, pp. 602-613, 2011.

[7] [7] S. M. Lundberg et al., "From Local Explanations to Global Understanding with Explainable AI for Trees," Nature Mach. Intell., vol. 2, no. 1, pp. 56-67, 2020.

[8] [8] H. He and E. A. Garcia, "Learning from Imbalanced Data," IEEE Trans. Knowl. Data Eng., vol. 21, no. 9, pp. 1263-1284, 2009.

[9] [9] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," Statistical Science, vol. 17, no. 3, pp. 235-255, 2002.

[10] [10] C. Phua, V. Lee, K. Smith, and R. Gayler, "A Comprehensive Survey of Data Mining-Based Fraud Detection Research," Artif. Intell. Rev., pp. 1-14, 2010.

[11] [11] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams, "Transaction Aggregation as a Strategy for Credit Card Fraud Detection," Data Min. Knowl. Discov., vol. 18, no. 1, pp. 30-55, 2009.

[12] [12] M. Brundage et al., "The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation," arXiv preprint arXiv:1802.07228, 2018.

[13] [13] C. Jiang et al., "A Mixed Deep Recurrent Neural Network for MOOC Dropout Prediction," IEEE Access, vol. 8, pp. 63441-63450, 2020.

[14] [14] Streamlit Inc., "Streamlit Documentation: Build and Share Data Apps," 2024. [Online]. Available: https://docs.streamlit.io/

[15] [15] SHAP Library, "SHAP (SHapley Additive exPlanations)," 2024. [Online]. Available: https://shap.readthedocs.io/

[16] [16] National Payments Corporation of India (NPCI), "UPI Product Statistics," 2024. [Online]. Available: https://www.npci.org.in/