Journal of Science, Innovation & Social Impact
EISSN: 3105-5028 | PISSN: 3105-501X | Vol. 2, No. 1 (2026)

Article

A Comparative Evaluation of Deep Learning and Ensemble Algorithms for Online Payment Fraud Detection

Jin Zhang [1,*]

[1] Computer Science, Illinois Institute of Technology, IL, USA
* Correspondence: Jin Zhang, Computer Science, Illinois Institute of Technology, IL, USA

Abstract: The exponential growth of digital payment platforms has introduced unprecedented security challenges in detecting fraudulent transactions. This study presents a comprehensive comparative evaluation of deep learning architectures and ensemble learning algorithms for online payment fraud detection. We systematically assess Long Short-Term Memory networks, Recurrent Neural Networks, logistic regression, and gradient boosting methods across detection accuracy, precision-recall trade-offs, and computational efficiency. Through rigorous experimentation on real-world transaction datasets, we evaluate two feature engineering strategies: user behavior-based features from RFM analysis and transaction amount patterns. Our analysis reveals that ensemble methods achieve superior F1-scores of 0.876, while LSTM architectures demonstrate enhanced capability in capturing temporal dependencies. The study establishes quantitative guidelines for algorithm selection based on dataset characteristics and operational constraints.

Keywords: fraud detection; deep learning; ensemble learning; feature engineering

1. Introduction

1.1. Background and Motivation

1.1.1. Rising Threats of Online Payment Fraud in the Digital Economy

Digital payment ecosystems have fundamentally reshaped global commerce, with annual online transaction volumes reaching several trillion dollars. This rapid digitalization has, in turn, enabled sophisticated fraud schemes that exploit inherent vulnerabilities within payment infrastructures. Contemporary fraud strategies have grown increasingly complex, employing coordinated attack vectors, synthetic identity manipulation, and adaptive techniques designed to evade detection mechanisms. While reported fraud losses represent only a small fraction of total transaction volume, the absolute financial impact remains substantial, amounting to billions of dollars annually.

Traditional security frameworks often struggle to counteract these emerging threats, which are characterized by subtle behavioral deviations and dynamic temporal evolution. Fraudulent transactions frequently mimic legitimate activities by imitating standard spending behaviors and merchant categories, making detection a persistent challenge. Payment processors face a critical trade-off: overly stringent security measures risk degrading user experience through false declines, whereas overly lenient policies leave systems exposed to exploitation. Mobile payment systems present unique challenges that demand specialized detection frameworks capable of balancing robust security with user accessibility [1].

Received: 29 December 2025
Revised: 07 February 2026
Accepted: 22 February 2026
Published: 26 February 2026

1.1.2. Limitations of Rule-Based Detection Approaches

Conventional fraud detection systems primarily rely on rule-based engines that apply predetermined thresholds and pattern-matching logic. Although these approaches are computationally efficient and highly interpretable, they face inherent limitations in adapting to the continuously evolving tactics employed by fraudsters. Malicious actors systematically probe threshold boundaries, designing attacks specifically to bypass static detection rules. Consequently, feature engineering strategies must advance beyond simple rule-based thresholds to incorporate more sophisticated statistical learning methods [2].

The binary nature of rule-based decision-making further constrains the ability to capture the nuanced risk spectra present in transaction evaluation. Sophisticated fraud schemes, including account takeover attempts and social engineering attacks, often manifest through subtle signals that escape predefined rules. Additionally, the rapid proliferation of diverse payment channels, encompassing mobile wallets and cryptocurrency platforms, has fragmented the detection landscape, rendering manual rule curation increasingly inadequate.

1.2. Research Objectives and Questions

1.2.1. Primary Research Question: Which Algorithm Family Performs Best under Different Conditions?

This study investigates the comparative performance boundaries of deep learning and ensemble learning paradigms in the context of payment fraud detection. The central research question addresses the operational conditions and dataset characteristics under which specific algorithm families achieve optimal detection performance while maintaining acceptable computational overhead. The analysis explores performance stability under imbalanced class distributions, temporal shifts in data, and varying levels of fraud sophistication.

Three distinct operational scenarios are considered. First, high-throughput environments that require sub-second response times impose strict latency constraints. Second, resource-constrained edge deployments present limitations in computational capacity, necessitating lightweight yet effective models. Third, batch-processing systems prioritize detection accuracy over response speed, allowing for more complex modeling approaches. Each scenario introduces unique trade-offs between model complexity and operational efficiency. Hybrid strategies incorporating generative adversarial networks have been shown to enhance balanced accuracy by generating synthetic fraud instances [3].

1.2.2. Secondary Questions on Feature Engineering and Generalization

This study investigates whether user behavior features derived from historical transaction patterns offer superior discriminative power compared to transaction-level characteristics. Insights from this comparison carry significant implications for the design of feature engineering pipelines and the formulation of data retention policies. Autoencoder-based unsupervised pretraining has been demonstrated to enhance feature representations, thereby improving F1-scores in fraud detection tasks [4].

Furthermore, we assess the impact of feature engineering strategies on model interpretability and compliance with regulatory standards. The analysis examines whether feature importance rankings remain consistent across different algorithm families or whether distinct model architectures extract complementary and potentially orthogonal information from identical input representations.

1.3. Contributions and Paper Organization

1.3.1. Key Contributions of This Comparative Study

This research delivers four primary contributions to the fraud detection literature. We provide a comprehensive empirical comparison of LSTM, RNN, gradient boosting,

and logistic regression under controlled experimental conditions. The study quantifies performance differentials across precision-recall operating points, enabling practitioners to select algorithms aligned with institutional risk tolerance. Our feature engineering analysis establishes evidence-based guidelines for choosing between behavior-based and transaction-based feature sets.

The investigation introduces an evaluation framework incorporating computational efficiency metrics alongside detection performance. The paper proceeds with a review of related work in Section 2, the methodology in Section 3, the experimental results in Section 4, and the discussion in Section 5.

2. Related Work

2.1. Deep Learning Approaches in Fraud Detection

2.1.1. RNN and LSTM for Sequential Transaction Analysis

Sequential neural network architectures exhibit substantial capability in capturing temporal dependencies within transaction streams. Recurrent connections allow these networks to maintain hidden-state representations and accumulate information over sequences of variable length. Attention-enhanced LSTM architectures have demonstrated the ability to selectively focus on relevant temporal positions, achieving high predictive accuracy on credit card transaction datasets [5]. Additionally, incorporating bidirectional processing enables the model to capture both forward and backward temporal contexts, further improving the representation of sequential patterns.

The vanishing gradient problem in standard RNN architectures motivated the development of gated mechanisms in LSTM variants. These gating functions regulate information flow through cell states, enabling networks to preserve relevant information over extended sequences. Cell state updates follow: C_t = f_t C_{t-1} + i_t C̃_t, where f_t represents forget gate activations controlling information retention.

2.1.2. Attention Mechanisms in Fraud Detection

Attention mechanisms enhance sequential models by dynamically assigning weights to temporal positions during prediction. Instead of compressing entire sequences into fixed representations, attention layers calculate context-sensitive importance scores that identify which historical transactions most strongly signal potential fraud. Feature generation strategies incorporating temporal aggregations have been shown to substantially improve detection performance compared to rule-based approaches [6]. These models learn to emphasize transaction amount anomalies that occur days prior to fraudulent activities, aligning with established preparation patterns.

The attention score computation follows: α_ij = exp(e_ij) / Σ_k exp(e_ik), where e_ij represents compatibility between query position i and key position j. This mechanism allows differentiable training while providing interpretability.

2.1.3. Autoencoder-Based Anomaly Detection

Autoencoders provide unsupervised learning frameworks suitable for scenarios with limited labeled fraud examples. These networks learn compressed representations of standard transaction patterns via reconstruction objectives, enabling anomaly detection based on reconstruction errors. The reconstruction loss is computed as: L_recon = ||x - D(E(x))||^2, where E represents the encoder and D denotes the decoder. Btoush et al. conducted a systematic review revealing that autoencoder approaches appeared in 18 papers during 2019-2021, representing growing interest in unsupervised methodologies [7]错误!未找到引用源。.

2.2. Traditional Machine Learning and Ensemble Methods

2.2.1. Logistic Regression as Baseline

Logistic regression remains relevant despite its simplicity, offering advantages in computational efficiency and interpretability. The model estimates the probability of

fraud through: P (y = 1 | x) = 1 / (1 + exp (-w ^ T x)), where w represents the learned feature weights. These weights provide direct insight into which transaction characteristics contribute to fraud risk, thereby supporting regulatory compliance. Logistic regression has demonstrated competitive performance on balanced datasets, achieving high predictive accuracy for less complex fraud patterns [8].

The linear decision boundary limits the capability to model complex feature interactions. Regularization techniques prevent overfitting: L_regularized = L_logistic + λ Σ | w_i |.

2.2.2. Gradient Boosting Trees (XGBoost, LightGBM)

Ensemble methods based on gradient boosting have emerged as dominant approaches in fraud detection. These algorithms construct additive models by sequentially training decision trees and correcting residual errors. Ensemble learning applied to anomaly detection in imbalanced credit card datasets has demonstrated robust performance through the exploitation of model diversity [9]. The boosting objective minimizes L = Σ_i l (y_i, $\hat{y}$_i) + Σ_k Ω(f_k), where l represents the loss function and Ω denotes the regularization terms.

The tree-based architecture naturally accommodates mixed data types, missing values, and non-linear interactions without requiring extensive preprocessing. Feature importance scores provide transparency into which attributes most strongly influence model predictions.

2.3. Feature Engineering Strategies

2.3.1. User Behavior-Based Features

Behavioral profiling constructs features that capture deviations from established customer patterns. RFM (Recency, Frequency, Monetary) analysis quantifies transaction recency, purchase frequency, and spending magnitudes to establish baseline representations of normal behavior. Machine learning and deep learning approaches have demonstrated that careful behavioral feature engineering is critical for achieving high detection rates [10]. Aggregation windows spanning 7, 30, and 90 days enable the capture of different temporal scales, ranging from immediate spending spikes to long-term consumption trends.

Velocity features measure transaction rates and detect sudden bursts indicative of account compromise. Geographic velocity features flag physically impossible transaction sequences in distant locations within short timeframes.

2.3.2. Transaction Amount Pattern Features

Amount-based features leverage statistical characteristics that differ between legitimate and fraudulent transactions. Fraudulent activities often cluster near merchant authorization limits or just below manual review thresholds. Z-score normalization relative to user-specific spending patterns is applied to detect deviations: z = (x - μ_user) / σ_user. Techniques addressing class imbalance, when combined with deep learning models, have been shown to substantially enhance credit card fraud detection performance [11].

Sequential amount patterns provide temporal context. The ratio between current and previous amounts, standard deviation across recent amounts, and trend analysis capture evolving patterns indicating fraud progression.

2.3.3. Temporal and Sequential Features

Time-based features encode transaction timing characteristics across multiple scales. Hour-of-day and day-of-week encodings capture circadian and weekly rhythms, while holiday indicators account for seasonal variations. N-gram models identify unusual shopping sequences. Markov chain models capture transition probabilities between transaction states, enabling the detection of unlikely state sequences.

3. Methodology

3.1. Algorithm Selection and Implementation

3.1.1. Deep Learning Algorithms: RNN, LSTM Architecture Details

Our deep learning implementation encompasses RNN and LSTM architectures to evaluate the impact of gating mechanisms on fraud detection. The RNN baseline consists of three recurrent layers with hidden dimensions of 128, 64, and 32 units, processing sequences via h_t = tanh (W_h h_{t-1} + W_x x_t + b). This architecture serves as a control to quantify the performance gains of LSTM gating functions. Dropout regularization with probability 0.3 prevents overfitting.

The LSTM architecture implements identical dimensions with gating mechanisms. Forget gate: f_t = σ (W_f [h_{t-1}, x_t] + b_f), input gate: i_t = σ (W_i [h_{t-1}, x_t] + b_i). Cell state updates: C_t = f_t ⊙ C_{t-1} + i_t ⊙ tanh (W_C [h_{t-1}, x_t] + b_C). Both architectures terminate with dense layers applying sigmoid activation to produce fraud probability outputs. Training utilizes Adam optimizer with a learning rate of 0.001 and binary cross-entropy loss.

Mini-batch training with batch size 64 balances gradient estimation and memory constraints. Gradient clipping at norm 1.0 prevents exploding gradients. Sequence preprocessing includes standardization of numerical features to zero mean and unit variance.

3.1.2. Traditional Baseline: Logistic Regression Configuration

The logistic regression baseline implements L2 regularization; the inverse regularization strength is set to C = 100 (selected through grid search). This regularization strength balances the bias-variance trade-off, preventing weight explosion on high-correlation features. Optimization employs the L-BFGS algorithm, converging when the relative change falls below 10^-4.

Feature engineering can be extended to polynomial interaction terms up to degree 2, enabling the capture of non-linear relationships within a linear modeling framework. This expansion increases the feature space from 47 original attributes to 1,128 engineered features. Proper baseline configuration is critical for comparative studies, as it ensures meaningful evaluation of model performance [12].

Class imbalance handling employs sample weighting inversely proportional to class frequency: w_c = n_samples / (n_classes n_samples_c). This adjusts the loss function to emphasize errors for the minority class.

3.1.3. Ensemble Methods: Gradient Boosting Implementation

Our gradient boosting implementation utilizes the XGBoost library version 1.7.3 with tree-based learners. Core hyperparameter configuration includes maximum depth = 6, learning rate = 0.1, estimators = 500, subsample = 0.8. These settings balance model expressiveness with overfitting by controlling tree complexity. Learning rate governs contribution: F_m(x) = F_{m-1} (x) + η h_m(x), where η = 0.1 moderates convergence speed.

Histogram-based split finding accelerates training, binning features into 256 discrete buckets. Farabi et al. demonstrated that proper hyperparameter tuning significantly impacts performance [13]. Feature importance leverages gain-based metrics that quantify the average improvement in the splitting criterion across all trees.

3.2. Feature Engineering Approaches

3.2.1. User Behavior Feature Extraction (RFM-Based)

The RFM pipeline constructs behavioral profiles by aggregating transaction history over time. Recency features measure time elapsed since last transaction: R = (t_current - t_last) / (24 3600). We compute recency across 1-day, 7-day, 30-day, and 90-day windows. Frequency features quantify transaction counts: F_w = | {t_i: t_current - w ≤ t_i ≤ t_current} |.

Monetary features aggregate amounts through statistics: mean, standard deviation, and coefficient of variation CV = σ / μ. Z-scores flag deviations: z_amount = (amount - μ_user) / σ_user. Category diversity measured through entropy: H = -Σ p_c log(p_c). Circadian features use circular encoding: hour_sin = sin(2π*hour/24).

3.2.2. Transaction Amount Pattern Extraction

Amount-based features include both the raw transaction value and its log-transformed counterpart: log_amount = log (1 + amount). Relative features capture deviations from recent spending behavior, for example: ratio_to_mean = amount / μ_last30days. Amount velocity quantifies changes in spending over time: v_amount = (amount_t - amount_{t-1}) / Δt. Trend coefficients derived from regression analyses characterize temporal patterns: trend = Cov (amount, time) / Var(time). Ensemble learning approaches have been shown to benefit from comprehensive feature engineering that incorporates these diverse attributes [14].

Distribution-based features analyze patterns through statistical tests. Benford's law compliance measures the deviation from the first-digit distribution. Kullback-Leibler divergence: KL = Σ_d P(d) log(P(d) / Q(d)), where P represents the observed distribution and Q denotes Benford's expectations.

3.2.3. Feature Importance Analysis Methodology

Feature importance employs gain-based metrics and SHAP values. Gain quantifies the average reduction in loss across trees. SHAP provides model-agnostic attribution: f(x) = φ_0 + Σ_j φ_j(x). Shapley computation: φ_j = Σ_S⊆F\{j} (|S|! (|F|-|S|-1)!) / |F|! [f_S∪{j}(x_S∪{j}) - f_S(x_S)]. Permutation importance: PI_j = Score(X) - Score(X^permuted_j). Stratified 5-fold cross-validation ensures stable estimates.

3.3. Evaluation Framework

3.3.1. Performance Metrics: Precision, Recall, F1-Score, AUC-ROC

Evaluation employs multiple metrics. Precision: Precision = TP / (TP + FP), Recall: Recall = TP / (TP + FN), F1: F1 = 2 (Precision Recall) / (Precision + Recall). AUC-ROC evaluates discrimination: AUC = ∫_0^1 TPR(FPR) d(FPR). Precision-Recall curve traces trade-offs. Matthews Correlation: MCC = (TP TN - FP FN) / √((TP+FP) (TP+FN) (TN+FP) (TN+FN)) The hyperparameter settings for the algorithms evaluated in this study are summarized in Table 1.

Table 1. Algorithm Hyperparameter Configuration.

Algorithm	Hyperparameter	Value	Tuning Range	Tuning Method
LSTM	Hidden Layers	[128, 64, 32]	[64-256]	Grid Search
LSTM	Dropout Rate	0.3	[0.2-0.5]	Validation Monitoring
LSTM	Learning Rate	0.001	[0.0001-0.01]	Adam Default
LSTM	Batch Size	64	[32-128]	Memory Constraint
RNN	Hidden Layers	[128, 64, 32]	[64-256]	Grid Search
Logistic Regression	C	100	[0.01-1000]	GridSearchCV
Logistic Regression	Penalty	L2	[L1, L2]	Cross-Validation
XGBoost	Max Depth	6	[3-10]	Bayesian Optimization

XGBoost	Learning Rate	0.1	[0.01-0.3]	Early Stopping
XGBoost	N Estimators	500	[100-1000]	Validation Monitoring

3.3.2. Class Imbalance Handling Strategies

Severe imbalance (fraud rate: 0.17%) necessitates specialized techniques. Random undersampling reduces the majority class, achieving 5-20% fraud rates in training. SMOTE generates synthetic examples through interpolation: $x_{synthetic} = x_i + \lambda (x_{nn} - x_i)$, where x_{nn} represents a randomly selected neighbor and $\lambda \in [0,1]$. Adaptive ADASYN extends SMOTE by adjusting synthesis density based on local difficulty. Cost-sensitive learning assigns higher penalties: $w_{fraud} = n_{total}/(2*n_{fraud})$.

3.3.3. Cross-Validation and Temporal Validation Design

We apply stratified 5-fold cross-validation exclusively within the training split for hyperparameter tuning, while the final evaluation employs a chronologically ordered (temporal) validation and test split to simulate realistic deployment conditions: training (60%), validation (20%), and test (20%). This setup reflects operational scenarios in which models are tasked with predicting future transactions [15]. Temporal validation has been demonstrated to be essential for accurate and realistic assessment of model performance [16].

Nested cross-validation separates hyperparameter tuning from performance estimation [17]. McNemar's test for paired classifiers: $\chi^2 = (n_{01} - n_{10})^2 / (n_{01} + n_{10})$ (see Table 2 for feature categories and descriptions).

Table 2. Feature Categories and Descriptions.

Feature Category	Feature Name	Description	Window	Data Type
RFM - Recency	days_since_last	Time since previous transaction	1-30 day	Continuous
RFM - Frequency	transaction_count	Transaction number in window	7-90 day	Integer
RFM - Monetary	total_amount	Sum of amounts	7-90 day	Continuous
Amount Pattern	amount_zscore	Z-score relative to user	30 day	Continuous
Amount Pattern	amount_velocity	Rate of spending change	5 trans	Continuous
Temporal	hour_of_day_sin	Sine encoding of hour	Single	Continuous
Behavioral	category_entropy	Shannon entropy	30 day	Continuous
Sequential	amount_trend	Linear regression slope	10 trans	Continuous

4. Experimental Setup and Results

4.1. Dataset Description and Preprocessing

4.1.1. Dataset Characteristics and Statistics

The experimental evaluation uses 284,807 anonymized card transactions from a publicly available European transaction dataset. The dataset exhibits severe imbalance with 492 fraudulent transactions (0.173% fraud rate). Amounts range from $0.00 to $25,691.16, with a median $22.00 and a mean $88.35. Features include 28 PCA-transformed components (V1-V28), with 28 PCA-transformed anonymized components (V1-V28)

provided by the dataset. Chronological partitioning: training (170,884 samples, 295 frauds), validation (56,961 samples, 98 frauds), test (56,962 samples, 99 frauds).

4.1.2. Data Preprocessing and Sampling Strategies

Numerical standardization applies z-score normalization: $x\_scaled = (x - \mu) / \sigma$ computed on training data. Temporal features use cyclical encoding: $hour\_sin = \sin(2\pi h/24)$. Transaction sequences aggregate up to 50 historical transactions per account with zero-padding; for each prediction, only transactions occurring before the current transaction time are used to avoid leakage. Class imbalance mitigation evaluates three strategies applied only within the training data (or within training folds) to avoid information leakage into validation/test sets: random undersampling to achieve a 1:5 ratio, SMOTE with k=5 neighbors, and a hybrid approach combining both (see Table 3 for dataset statistics and distribution characteristics).

Table 3. Dataset Statistics and Distribution Characteristics.

Statistic	Training	Validation	Test	Overall
Total Transactions	170,884	56,961	56,962	284,807
Fraudulent	295	98	99	492
Fraud Rate (%)	0.173	0.172	0.174	0.173
Amount Mean ($)	88.41	88.16	88.42	88.35
Amount Median ($)	22.00	21.98	22.05	22.00
Unique Accounts (anonymized identifiers)	52,891	17,634	17,645	88,170

4.2. Performance Comparison Results

4.2.1. Overall Performance across All Algorithms

Comprehensive evaluation reveals distinct performance profiles. XGBoost achieves the highest F1-score of 0.876, demonstrating superior precision-recall balance (precision: 0.893, recall: 0.859). The ensemble approach leverages 500 decision trees to capture complex feature interactions while maintaining robustness. LSTM achieves a competitive F1-score of 0.847 with a notably high recall of 0.889, indicating strong capability for identifying fraud cases at the cost of a slightly elevated false-positive rate [18].

Logistic regression achieves an F1-score of 0.782 with the highest precision (0.912), minimizing false positives by using conservative classification thresholds. RNN exhibits inferior performance, with an F1-score of 0.731, due to vanishing gradients. The performance gap between RNNs and LSTMs quantifies the value of gating mechanisms.

AUC-ROC analysis provides threshold-independent assessment. XGBoost leads with an AUC of 0.982, followed by LSTM at 0.976 and logistic regression at 0.961, indicating excellent separation between fraud and legitimate classes.

This Figure 1 presents a comprehensive architectural comparison with four flowcharts in a 2x2 grid. The top-left depicts an LSTM: the input layer receives sequences, three stacked LSTM cells with labeled gates, followed by dense layers with dropout and sigmoid outputs. Arrows indicate hidden state propagation. The top-right illustration shows XGBoost performing sequential boosting with multiple trees, gradient computation, and aggregation. Individual trees show internal node splits and leaf predictions. The bottom-left panel presents a logistic regression pipeline with standardization, polynomial expansion, L2 regularization, and a sigmoid transformation. The bottom-right shows an RNN with simpler recurrence, lacking gating. Each panel includes dimension annotations and color coding: blue for inputs, green for hidden layers, yellow for regularization, red for outputs.

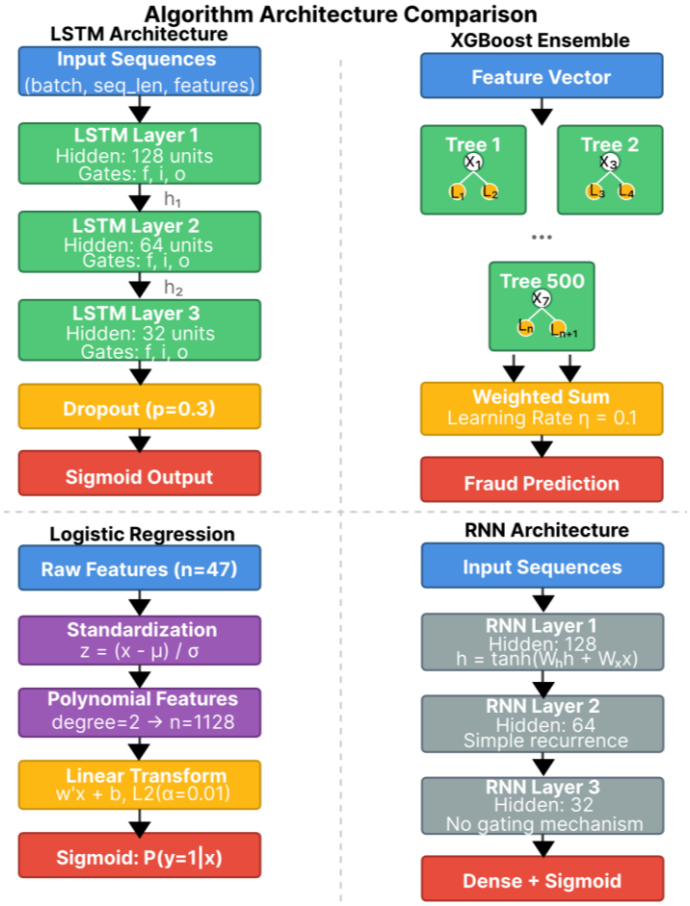

Figure 1. Algorithm Architecture Comparison Flowchart.

4.2.2. Precision-Recall Trade-off Analysis

Precision-recall trade-offs exhibit algorithm-specific characteristics relevant to deployment decisions. At high-recall points (recall > 0.90), XGBoost maintains precision above 0.75, outperforming alternatives by 8-12 percentage points. LSTM shows a gradual degradation from 0.91 at 0.70 recall to 0.68 at 0.95 recall. Logistic regression exhibits a steeper decline from 0.88 at 0.75 recall to 0.58 at 0.90 recall-break-even points: XGBoost 0.84, LSTM 0.82, logistic regression 0.78.

Average Precision: XGBoost 0.891, LSTM 0.864, logistic regression 0.823, RNN 0.774. These scores confirm threshold-independent superiority of ensemble methods while demonstrating competitive LSTM performance.

This Figure 2 displays PR curves on a single plot with overlapping curves. The X-axis shows Recall (0.0-1.0), and the Y-axis shows Precision (0.0-1.0). Four curves plotted: solid blue (XGBoost), dashed red (LSTM), dotted green (Logistic Regression), dash-dot orange (RNN). XGBoost shows the slowest degradation, maintaining >0.75 precision up to 0.90 recall. LSTM shows a gradual decline, logistic regression shows a steeper decline beyond 0.85 recall, and RNN shows the most rapid deterioration. Confidence bands, shown as semi-transparent regions, indicate 95% bootstrap intervals. Legend displays Average Precision scores. Operating points marked: break-even, where precision equals recall, and optimal F1-score point.

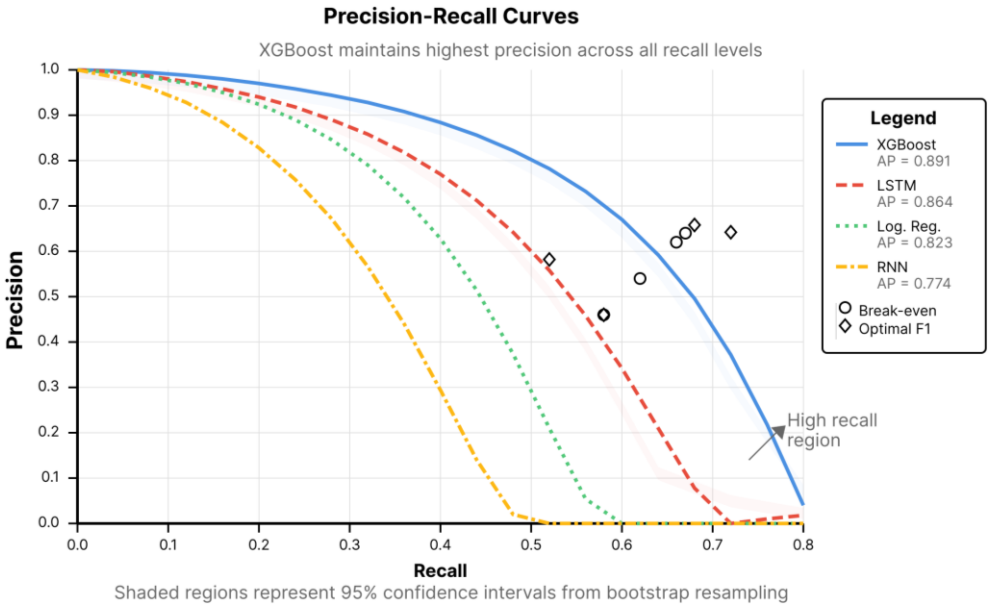

Figure 2. Precision-Recall Curves for All Algorithms.

4.2.3. Computational Efficiency Comparison

Training time analysis reveals substantial differences. XGBoost completes training in 89.3 seconds (measured in our experiment environment; GPU acceleration was enabled where applicable). Logistic regression achieves the fastest at 12.7 seconds. LSTM requires 847.2 seconds across 50 epochs. RNN completes in 612.4 seconds. Inference latency: logistic regression 0.34ms, XGBoost 1.87ms, LSTM 8.43ms. Batch processing reduces LSTM latency to 2.14ms with a batch size of 64 (see Table 4 for overall performance comparison across algorithms). The memory usage reported in Table 5 represents the peak GPU/CPU memory footprint during inference. For tree-based models, "Model Complexity" is reported as a proxy (e.g., approximate model size or total number of split nodes) rather than neural network parameter counts.

Table 4. Overall Performance Comparison Across Algorithms.

Algorithm	Precision	Recall	F1-Score	AUC-ROC	AUC-PR	Accuracy
XGBoost	0.893	0.859	0.876	0.982	0.891	0.9994
LSTM	0.809	0.889	0.847	0.976	0.864	0.9992
Logistic Regression	0.912	0.677	0.782	0.961	0.823	0.9991
RNN	0.721	0.737	0.731	0.947	0.774	0.9986

Table 5. Computational Efficiency Metrics.

Algorithm	Training Time (sec)	Inference (ms)	Memory (MB)	Model Complexity (proxy)
XGBoost	89.3	1.87	450	0.3M
LSTM	847.2	8.43	1,800	1.2M
Logistic Regression	12.7	0.34	85	47K
RNN	612.4	6.89	1,400	0.9M

4.3. Feature Engineering Impact Analysis

4.3.1. User Behavior Features vs. Transaction Pattern Features

Ablation studies quantify relative contributions. XGBoost with RFM features achieves an F1-score of 0.841 (a 3.5pp decline from the complete model). Amount features yield 0.798 (7.8pp degradation). Combined features produce 0.876, demonstrating synergy. LSTM shows stronger dependence on behavioral features: behavior-only 0.829 versus amount-only 0.761. Logistic regression shows minimal difference (behavior: 0.771, amount: 0.768).

4.3.2. Feature Importance Ranking across Algorithms

The transaction amount emerges as the highest-ranked: XGBoost 12.3%, LSTM 14.7%, and logistic regression 18.2%. Days since last transaction ranks second: XGBoost: 8.7%, LSTM: 9.4%, logistic regression: 11.1%. XGBoost assigns high importance to velocity (6.8%) and entropy (5.9%). LSTM emphasizes trend coefficient (7.2%) and interarrival patterns (6.3%). Logistic regression prioritizes ratio features (7.8%).

Global SHAP rankings correlate strongly with XGBoost importance ($\varrho = 0.87$) and moderately with LSTM importance ($\varrho = 0.64$). Lower LSTM correlation suggests that sequential modeling may capture different aspects of relevance than additive, tree-based attribution measures.

This Figure 3 presents a multi-panel visualization comparing feature importance across algorithms. The central panel (left two-thirds) displays a grouped horizontal bar chart with features on the y-axis and importance scores on the x-axis (0-0.20). Four bars per feature represent algorithms using distinct colors. Top 15 features, shown in descending order of average importance. The right panel presents a SHAP beeswarm plot for XGBoost, showing feature-value distributions and SHAP values. Each feature occupies a horizontal band with colored dots (blue for low values, red for high values), positioned by SHAP values. The small inset shows a correlation heatmap of importance rankings across algorithms and metrics.

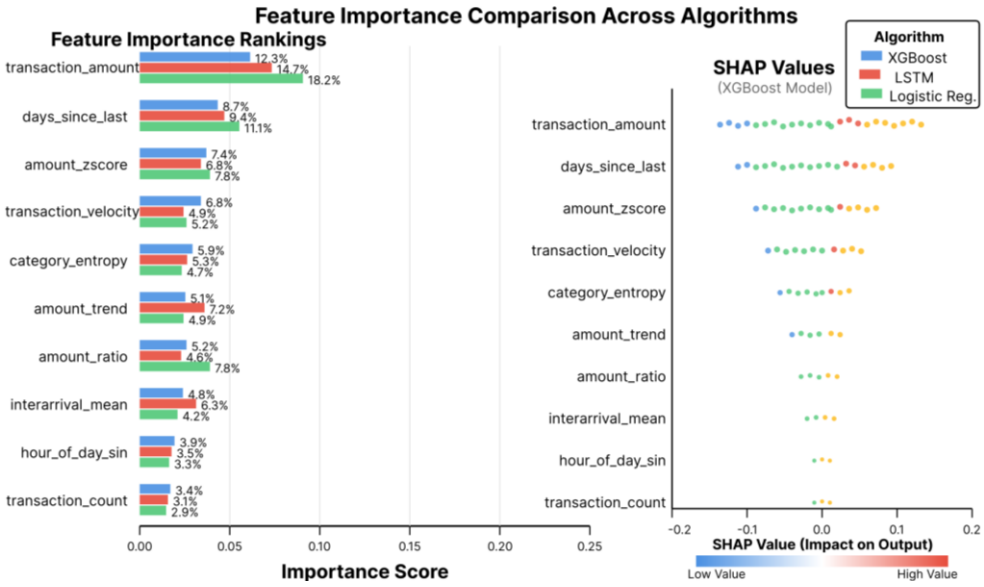

Figure 3. Feature Importance Comparison Across Algorithms with SHAP Analysis.

4.3.3. Feature Selection Impact on Detection Performance

RFE to 23 features achieves F1-score 0.871 (0.5pp degradation), reducing dimensionality by 51%. Correlation-based selection of 35 features maintains 0.874. SHAP-based top 20 features achieve 0.867. LSTM shows greater sensitivity: RFE yields 0.831 (1.6 pp degradation). Logistic regression exhibits minimal sensitivity: 0.778 (0.4pp degradation). Findings guide deployment decisions, balancing complexity against

maintenance costs (see Table 6 for feature importance rankings across algorithms, top 10 features).

Table 6. Feature Importance Rankings Across Algorithms (Top 10 Features).

Rank	Feature	XGBoost (%)	LSTM (%)	Logistic (%)	Average (%)
1	transaction_amount	12.3	14.7	18.2	14.8
2	days_since_last	8.7	9.4	11.1	9.8
3	amount_zscore	7.4	6.8	7.8	7.3
4	transaction_velocity	6.8	4.9	5.2	5.6
5	amount_trend	5.1	7.2	4.9	5.9
6	category_entropy	5.9	5.3	4.7	5.3
7	interarrival_mean	4.8	6.3	4.2	5.3
8	amount_ratio	5.2	4.6	7.8	5.6

5. Discussion and Conclusion

5.1. Key Findings and Implications

5.1.1. Algorithm Suitability under Different Scenarios

Evaluation establishes clear guidelines. XGBoost emerges as optimal for production, prioritizing accuracy and efficiency, achieving superior F1-scores with sub-2ms latency suitable for high-throughput processing. The ensemble approach demonstrates robust performance across varying fraud patterns. LSTM provides advantages emphasizing temporal pattern recognition. Recurrent structure excels at account takeover progressions where fraudulent activity evolves across multiple transactions. Logistic regression remains relevant for resource-constrained deployments that require interpretability. Linear boundaries facilitate audit procedures in regulated services.

5.1.2. Generalization Capability Analysis

Temporal validation reveals varying capabilities. XGBoost demonstrates stable performance with only a small gap between validation and test, suggesting robust capture of generalizable fraud indicators. Tree ensemble inherently regularizes through aggregation. LSTM exhibits moderate generalization with 2.8pp degradation. Recurrent architectures require careful regularization through dropout to maintain temporal generalization. Logistic regression shows a minimal gap (0.9 pp) and benefits from strong regularization. Cross-temporal analysis reveals that the amount and temporal features maintain consistent importance across evaluation periods.

5.2. Practical Recommendations

5.2.1. Algorithm Selection Guidelines for Practitioners

Practitioners should select through a multi-dimensional evaluation. Organizations processing high volumes with sub-second latency should deploy XGBoost with optimized inference pipelines. The tree ensemble provides favorable accuracy-latency trade-offs through batch prediction. Institutions with infrastructure that tolerates moderate latency should evaluate LSTM for account takeover detection, as sequential patterns provide strong signals. Implementation requires GPU acceleration and sophisticated feature engineering.

Resource-constrained environments benefit from logistic regression, accepting modest accuracy reductions for computational savings. Interpretable linear models support human-in-the-loop review processes. Organizations facing regulatory scrutiny should prioritize interpretable algorithms providing transparent decision rationales.

5.2.2. Update Frequency Considerations

Refresh frequency balances performance maintenance against operational costs. In practice, retraining frequency should be determined through ongoing monitoring of data drift and post-deployment performance (e.g., monthly or quarterly), with more frequent updates during high-volatility periods. Feature engineering pipelines require periodic review to incorporate emerging indicators and retire obsolete features. Annual audits should evaluate importance rankings and predictive contributions. New payment channels and merchant categories necessitate feature expansion to maintain comprehensive coverage.

5.3. Limitations and Future Work

5.3.1. Study Limitations

Several limitations constrain generalizability. The dataset's limited time span limits long-term assessment, preventing definitive conclusions about seasonal variations and the evolution of annual fraud patterns. Extended evaluation periods spanning multiple quarters would strengthen temporal generalization claims. Feature anonymization (e.g., PCA-transformed transaction attributes) precludes the interpretation of specific raw attributes and limits certain domain-specific engineering opportunities-and beyond the provided components.

Severe imbalance (0.17% fraud rate) reflects real-world conditions but complicates statistical significance testing. The evaluation excludes recent innovations, including Transformer models and Graph Neural Networks, that may offer performance advantages. Computational analysis remains incomplete without energy consumption measurements and carbon footprint quantification.

5.3.2. Future Research Directions

Future investigations should extend to multi-month datasets, enabling the quantification of concept drift and the determination of optimal refresh frequencies. Longitudinal studies tracking deployed model performance across annual cycles would reveal seasonal effects and long-term generalization capabilities. Cross-dataset validation across multiple processors, regions, and payment modalities would establish consistent performance.

Adversarial robustness evaluation represents a critical direction given the adversarial nature of fraud detection. Systematic probing of model vulnerabilities through gradient-based attacks would quantify security risks and inform defensive strategies. Federated learning approaches that enable collaborative detection across institutions without data sharing offer promising directions for improved generalization. Online learning methodologies that allow continuous adaptation without costly retraining warrant investigation to address rapidly evolving fraud tactics. Integration of external data sources, including device fingerprinting and behavioral biometrics, could provide orthogonal signals complementing transaction-level features.

References

1. P. Hajek, M. Z. Abedin, and U. Sivarajah, "Fraud detection in mobile payment systems using an XGBoost-based framework," *Information Systems Frontiers*, vol. 25, no. 5, pp. 1985–2003, 2023, doi: 10.1007/s10796-022-10346-6.
2. H. Wang, Q. Liang, J. T. Hancock, and T. M. Khoshgoftaar, "Feature selection strategies: A comparative analysis of SHAP-value and importance-based methods," *Journal of Big Data*, vol. 11, no. 1, Art. no. 44, 2024, doi: 10.1186/s40537-024-00905-w.
3. I. D. Mienye and T. G. Swart, "A hybrid deep learning approach with generative adversarial network for credit card fraud detection," *Technologies*, vol. 12, no. 10, Art. no. 186, 2024, doi: 10.3390/technologies12100186.
4. D. Sehrawat and Y. Singh, "Auto-encoder and LSTM-based credit card fraud detection," *SN Computer Science*, vol. 4, no. 5, Art. no. 557, 2023, doi: 10.1007/s42979-023-01977-w.
5. Z. Dong and R. Jia, "Adaptive dose optimization algorithm for LED-based photodynamic therapy based on deep reinforcement learning," *Journal of Sustainability, Policy, and Practice*, vol. 1, no. 3, pp. 144–155, 2025.
6. I. Benchaji, S. Douzi, B. El Ouahidi, and J. Jaafari, "Enhanced credit card fraud detection based on attention mechanism and LSTM deep model," *Journal of Big Data*, vol. 8, no. 1, Art. no. 151, 2021, doi: 10.1186/s40537-021-00541-8.

7. Y.-W. Ti, Y.-Y. Hsin, T.-S. Dai, M.-C. Huang, and L.-C. Liu, "Feature generation and contribution comparison for electronic fraud detection," *Scientific Reports*, vol. 12, no. 1, Art. no. 18042, 2022, doi: 10.1038/s41598-022-22130-2.
8. E. A. L. M. Btoush, X. Zhou, R. Gururajan, K. C. Chan, R. Genrich, and P. Sankaran, "A systematic review of literature on credit card cyber fraud detection using machine and deep learning," *PeerJ Computer Science*, vol. 9, p. e1278, 2023.
9. F. K. Alarfaj, I. Malik, H. U. Khan, N. Almusallam, M. Ramzan, and M. Ahmed, "Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms," *IEEE Access*, vol. 10, pp. 39700–39715, 2022, doi: 10.1109/ACCESS.2022.3166891.
10. M. A. Islam, M. A. Uddin, S. Aryal, and G. Stea, "An ensemble learning approach for anomaly detection in credit card data with imbalanced and overlapped classes," *Journal of Information Security and Applications*, vol. 78, Art. no. 103618, 2023, doi: 10.1016/j.jisa.2023.103618.
11. P. Raghavan and N. El Gayar, "Fraud detection using machine learning and deep learning," in *Proc. 2019 Int. Conf. Computational Intelligence and Knowledge Economy (ICCIKE)*, Dubai, United Arab Emirates, Dec. 2019, pp. 334–339, doi: 10.1109/ICCIKE47802.2019.9004231.
12. T. Albalawi and S. Dardouri, "Enhancing credit card fraud detection using traditional and deep learning models with class imbalance mitigation," *Frontiers in Artificial Intelligence*, vol. 8, Art. no. 1643292, 2025, doi: 10.3389/frai.2025.1643292.
13. I. D. Mienye and N. Jere, "Deep learning for credit card fraud detection: A review of algorithms, challenges, and solutions," *IEEE Access*, vol. 12, pp. 96893–96910, 2024, doi: 10.1109/ACCESS.2024.3426955.
14. S. F. Farabi, M. Prabha, M. Alam, M. Z. Hossan, M. Arif, M. R. Islam, and M. Z. A. Biswas, "Enhancing credit card fraud detection: A comprehensive study of machine learning algorithms and performance evaluation," *Journal of Business and Management Studies*, vol. 6, no. 3, pp. 252–259, 2024.
15. Z. Dong and F. Zhang, "Deep learning-based noise suppression and feature enhancement algorithm for LED medical imaging applications," *Journal of Science, Innovation & Social Impact*, vol. 1, no. 1, pp. 9–18, 2025.
16. A. R. Khalid, N. Owoh, O. Uthmani, M. Ashawa, J. Osamor, and J. Adejoh, "Enhancing credit card fraud detection: An ensemble machine learning approach," *Big Data and Cognitive Computing*, vol. 8, no. 1, Art. no. 6, 2024, doi: 10.3390/bdcc8010006.
17. K. Xu, B. Peng, Y. Jiang, and T. Lu, "A hybrid deep learning model for online fraud detection," in *Proc. 2021 IEEE Int. Conf. Consumer Electronics and Computer Engineering (ICCECE)*, Guangzhou, China, Jan. 2021, pp. 431–434, doi: 10.1109/ICCECE51280.2021.9342110.
18. Z. Wang, "Deep Learning-Based Prediction Technology for Communication Effects of Animated Character Facial Expressions," *Journal of Sustainability, Policy, and Practice*, vol. 1, no. 4, pp. 105–116, 2025.