Journal of Computer Science

Research Article

Comparative Analysis of Fraud Detection Methods in Banking Using Machine Learning Techniques

Youssef Tounsi[1] and Ennouri Tazi[2]

[1]*RITM Laboratory, CED Engineering Sciences, Ecole Supérieure de Technologie, Hassan II University of Casablanca, Morocco*
[2]*Department of Economic Science and Management, FSJES Ain Chock, LIDSI Laboratory, Hassan II University of Casablanca, Morocco*

Article history
Received: 01-10-2025
Revised: 05-03-2026
Accepted: 22-05-2026

Corresponding Author:
Youssef Tounsi
RITM Laboratory, CED Engineering Sciences, Ecole Supérieure de Technologie, Hassan II University of Casablanca, Morocco
Email: tounsi@gmail.com

Abstract: Fraud detection in banking requires algorithms that balance classification performance, computational efficiency, and regulatory interpretability, criteria that are rarely benchmarked together. We present a comprehensive evaluation of nine machine learning approaches (Logistic Regression, Decision Tree, Random Forest, SVM, SGD, XGBoost, CatBoost, LightGBM, and MLP) across three datasets differing in size, imbalance severity, and feature type (synthetic, real-world PCA-anonymized, and large-scale simulated). Our protocol addresses four methodological gaps prevalent in the literature: (1) SMOTE applied strictly within cross-validation folds to prevent data leakage; (2) Systematic reporting of confidence intervals for all metrics; (3) Systematic inference latency profiling; and (4) SHAP-based interpretability analysis aligned with regulatory requirements. SHAP analysis provides model-agnostic feature attributions aligned with regulatory explainability requirements. Results show gradient boosting methods achieving superior fraud detection (CatBoost: F1 = 0.86 on real-world data) with sub-6ms inference, while SVM is disqualified for production use due to O(n2) latency scaling. This study provides reproducible baselines, with full hyperparameter specifications, to support algorithm selection in operational fraud prevention systems.

Keywords: Banking Fraud Detection, Machine Learning, Gradient Boosting, Class Imbalance, SHAP, Model Interpretability, Real-Time Inference, Deep Learning, Cross-Validation

Introduction

The digitalization of financial services has transformed banking operations while simultaneously creating unprecedented attack surfaces for fraudulent activities. Payment card fraud losses reached USD 33.83 billion globally in 2023, with cumulative losses projected to exceed USD 400 billion over the next decade (Report, 2025). Financial institutions bear a disproportionate share of this burden: The rapid expansion of Card-Not-Present transactions and e-commerce channels has fundamentally altered the threat landscape, rendering static rule-based defences increasingly inadequate (Ryman-Tubb et al., 2018; Al-Hashedi and Magalingam, 2021).

Traditional rule-based fraud detection systems while effective in controlled, static environments suffer from three fundamental limitations:

1) Inability to adapt to novel fraud patterns without manual rule updates
2) High false positive rates that disrupt legitimate customer activity
3) Scalability constraints when processing millions of daily transactions

Machine Learning (ML) addresses these limitations through adaptive pattern recognition, automated feature discovery, and probabilistic risk scoring. However, the ML literature on fraud detection is fragmented, and many benchmarking studies report results without confidence intervals or fail to control for data leakage during cross-validation, leading to overly optimistic performance claims (Dal Pozzolo et al., 2018; Carcillo et al., 2021). Furthermore, the literature rarely addresses inference latency, which is as critical as classification accuracy for real-time fraud prevention systems (Bahnsen et al., 2016).

This study addresses these gaps through rigorous experimental design and comprehensive reporting. The three datasets were deliberately selected to represent a spectrum of real-world fraud scenarios:

Dataset 1: (synthetic, 8.7% fraud rate, interpretable features) provides an upper-bound benchmark
Dataset 2: (real-world, 0.172% fraud rate, PCA-anonymized) represents a challenging production scenario
Dataset 3: (large-scale synthetic, 0.58% fraud rate, 22 features) tests computational scalability

This diversity enables assessment of algorithm robustness across fundamentally different data characteristics.

The primary contributions of this research are:

1. Rigorous comparative evaluation of nine ML approaches across three heterogeneous fraud datasets using stratified 5-fold cross-validation with SMOTE applied exclusively within training folds to prevent data leakage
2. Transparent investigation and explanation of near-perfect F1 scores, distinguishing genuine discriminability from potential data leakage artifacts
3. Inference latency benchmarking across all algorithms, providing actionable guidance for real-time deployment
4. SHAP-based feature importance analysis linking model decisions to interpretable fraud indicators for regulatory compliance
5. Explicit hyperparameter specifications and reproducibility information to facilitate future research

Literature Review

Evolution of Fraud Detection in Banking

The history of automated fraud detection spans four decades, progressing from expert systems and decision trees in the 1990s to contemporary deep learning architectures. Early systems relied on deterministic rules derived from domain expertise, flagging transactions exceeding predefined thresholds or occurring in unusual locations. While interpretable and auditable, these systems proved brittle against adaptive fraudsters (Ryman-Tubb et al., 2018).

The seminal work of Dal Pozzolo et al. (2018) established foundational benchmarks on the European credit card dataset, demonstrating that calibrated probability outputs from ensemble methods significantly outperform raw score thresholding in operational settings. Carcillo et al. (2021) subsequently demonstrated that combining unsupervised anomaly detection with supervised classifiers improves recall on novel fraud patterns. Al-Hashedi and Magalingam (2021) reviewed ML studies in banking fraud, finding that ensemble methods particularly gradient boosting variants dominate top-performing systems across diverse datasets.

Taxonomy of Banking Fraud

Banking fraud manifests across multiple attack vectors. Card-Not-Present (CNP) fraud exploits remote transaction channels and accounts for the majority of card fraud losses in high chip-adoption markets. Contactless payment fraud leverages relay attacks against NFC-enabled terminals. Card skimming involves physical compromise of magnetic stripe data at ATMs or point-of-sale terminals. Identity theft and synthetic identity fraud require cross-referencing with bureau data, while internal fraud perpetrated by banking personnel necessitates peer-group analysis (Ahmed et al., 2016; Randhawa et al., 2018).

Machine Learning in Fraud Detection: Critical Survey

Supervised learning remains dominant in production fraud detection systems due to its interpretability and availability of labeled historical data. Logistic regression remains a strong baseline providing calibrated probability outputs valued by compliance teams (Kim et al., 2019). Gradient boosting variants (XGBoost, LightGBM, CatBoost) have become the preferred approach for tabular fraud data, consistently achieving state-of-the-art results in both academic benchmarks and production deployments (Chen and Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018).

Deep learning has gained traction for sequential transaction modeling. Fu et al. (2016) demonstrate convolutional neural networks capturing temporal fraud patterns missed by tabular models. Forough and Momtazi (2021) show ensemble deep sequential networks excelling at account takeover detection. Xie et al. (2023) apply transformer-based attention architectures to transaction sequences, achieving superior performance when sequential context is available. Federated learning enables privacy-preserving collaborative fraud detection across institutions (Awosika et al., 2024). Fiore et al. (2019) propose generative adversarial networks for synthetic fraud example generation, demonstrating that augmentation strategies can complement classification algorithms.

Despite these advances, a persistent gap in the literature is the failure to report inference latency alongside classification metrics (Bahnsen et al., 2016), to apply SMOTE strictly within cross-validation folds (leading to data leakage), and to provide confidence intervals for reported scores. This study directly addresses all three gaps. Compared to prior comparative studies such as Dal Pozzolo et al. (2018); Carcillo et al. (2021), the present work extends the evaluation to nine algorithms, three datasets, and incorporates both latency

profiling and SHAP interpretability representing a more comprehensive operational benchmark.

Methods

Dataset Description and Justification

Three datasets were selected to represent a spectrum of real-world fraud detection scenarios, varying in size, class imbalance severity, feature interpretability, and data origin. Table 1 summarizes their key characteristics.

Dataset 1: Synthetic Credit Card Fraud

This dataset comprises 1,000,000 synthetic transactions with a fraud rate of 8.7%, the least severe imbalance in our benchmark. It contains seven highly discriminative features including distance from home, distance from last transaction, ratio to median purchase price, and binary indicators for chip usage, PIN usage, repeat retailer, and online order. Its interpretable features and relatively balanced class distribution make it ideal for evaluating algorithm ceiling performance. Near-perfect scores on this dataset reflect the clean synthetic decision boundaries rather than methodological artifacts, as confirmed by held-out test validation.

Dataset 2: Real-World Credit Card Fraud Detection

Contributed by the Machine Learning Group at Université Libre de Bruxelles, this dataset contains 284,807 real European credit card transactions with 492 fraudulent transactions (0.172% fraud rate). Features V1–V28 are principal components from PCA transformation, with only Time and Amount retained in original form. The extreme class imbalance and PCA-anonymized features make this dataset representative of real-world challenges. It is the most widely benchmarked fraud dataset in the literature, enabling direct comparison with prior work (Dal Pozzolo et al., 2018; Carcillo et al., 2021).

Dataset 3: Realistic Synthetic Transaction Data

This large-scale dataset contains 6,362,620 synthetic financial transactions with a fraud rate of 0.58%, It contains realistic synthetic transaction data. It includes 22 features spanning temporal, geographical, demographic, and transactional dimensions, enabling comprehensive feature engineering evaluation while testing computational scalability.

This dataset contains transaction records with various attributes that help in identifying fraudulent activities. The columns cover a mix of numerical, categorical, and temporal data, allowing for comprehensive fraud detection analysis.

Feature Engineering

Comprehensive feature engineering was conducted on Datasets 1 and 3. Temporal features derived from timestamps include hour of day, day of week, and transaction frequency within rolling windows. Behavioral deviation features capture departures from historical patterns:

$$\text{amount\_deviation} = \frac{|\text{amount} - \mu_{\text{historical}}|}{\sigma_{\text{historical}}}$$

$$\text{merchant\_consistency} = \frac{\text{merchant\_category\_frequency}}{\text{total\_transactions}}$$

Geographical features quantify spatial anomalies using the Haversine formula for distance between consecutive transactions and a location regularity score.

Data Preprocessing and Imbalance Handling

Continuous features were standardized using z-score normalization computed exclusively on training data to prevent leakage:

$$z = \frac{x - \mu_{\text{train}}}{\sigma_{\text{train}}}$$

To prevent data leakage in class imbalance handling a common but under-documented source of inflated performance estimates SMOTE and random under-sampling were applied exclusively within each training fold, after fold splitting. Within each training fold:

(1) SMOTE oversampled the minority class to a 1:5 ratio
(2) Random under-sampling achieved a 1:2 final ratio for Datasets 2 and 3
(3) Class weight parameters were additionally set in tree-based models

Table 1: Dataset Characteristics and Justification

Dataset	Source	Size	Fraud Rate	Features	Purpose
DS1: Synthetic CC Fraud	Kaggle	1,000,000	8.7%	7 features (distance, chip, PIN, amount ratio…)	Performance benchmark / https://www.kaggle.com/datasets/dhanushnarayananr/credit-card-fraud
DS2: Real CC Fraud (ULB)	Kaggle	284,807	0.172%	V1–V28 (PCA) + Time + Amount	Real-world extreme imbalance scenario / https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
DS3: Realistic synthetic transaction data	Kaggle	6,362,620	0.58%	22 features (temporal, geographic, balance)	Large-scale scalability evaluation / https://www.kaggle.com/datasets/samayashar/fraud-detection-transactions-dataset

Validation folds retained their original class distributions throughout. Stratified 5-fold cross-validation ensured representative fraud class proportions in all folds:

$$\mathrm{CV}_{\mathrm{score}} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{score}\left(M_i, D_{\mathrm{test}}^{(i)}\right)$$

Machine Learning Algorithms

We provide mathematical formulations and algorithm-specific implementation details for reproducibility. Full hyperparameter specifications are provided in Table 2.

Logistic Regression

Logistic Regression, first introduced by Cox (1958), represents one of the foundational algorithms in statistical machine learning for binary classification problems. In the context of fraud detection, logistic regression has been extensively studied and proven effective for modeling the probability of fraudulent transactions. The algorithm transforms linear combinations of input features through the sigmoid function to produce probability estimates between 0 and 1. This approach is particularly valuable in fraud detection as it provides interpretable probability scores that can be used for risk assessment and decision-making processes in banking environments:

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^{\top}\mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^{\top}\mathbf{x}+b)}}$$

Where: y is the binary target variable (0: legitimate, 1: fraud); x is the feature vector representing transaction characteristics; w is the weight vector learned during training; b is the bias term accounting for class prior probabilities; and σ is the sigmoid function mapping real values to $[0,1]$.

The optimization objective uses maximum likelihood estimation, implemented through the logistic loss function:

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i log\left(h_w(x_i)\right) + (1 - y_i) log\left(1 - h_w(x_i)\right)\right] + \frac{\lambda}{2} \parallel w \parallel^2$$

Despite its linear decision boundary, logistic regression serves as an essential interpretability baseline: Coefficients provide direct feature attribution satisfying regulatory explainability requirements. The algorithm's main advantages include computational efficiency, interpretability of coefficients, and robustness to outliers when properly regularized. However, its linear decision boundary limits performance on datasets with complex non-linear relationships. class_weight='balanced' adjusts decision boundaries for imbalanced data.

Decision Trees (CART)

Classification and Regression Trees (CART), originally developed by Breiman et al. (2017), represent a fundamental non-parametric machine learning approach that has found extensive application in fraud detection systems.

The algorithm's intuitive tree-like structure makes it particularly valuable in regulated financial environments where model interpretability is crucial. Decision trees operate by recursively partitioning the feature space into homogeneous regions through binary splits, creating interpretable rules directly aligned with existing fraud detection practices in banking institutions.

Table 2: Hyperparameter Specifications for All Algorithms

Algorithm	Key Hyperparameters	Tuning Strategy
Logistic Reg.	C=1.0, solver='lbfgs', max_iter=1000, class_weight='balanced'	Grid search: C ∈ {0.01, 0.1, 1, 10}
Decision Tree	max_depth=10, min_samples_split=20, criterion='gini', class_weight='balanced'	Grid search: depth ∈ {5, 10, 15, None}
Random Forest	n_estimators=200, max_depth=15, max_features='sqrt', class_weight='balanced_subsample'	RandomizedSearchCV, 50 iterations
SVM	C=10, kernel='rbf', gamma='scale', class_weight='balanced'	Grid search on C ∈ {1,10,100} and gamma
SGD	loss='log_loss', alpha=1e-4, learning_rate='optimal', class_weight='balanced'	Default with adaptive LR decay
XGBoost	n_estimators=300, max_depth=6, lr=0.05, scale_pos_weight=ratio, subsample=0.8, colsample_bytree=0.8	Bayesian opt. (Optuna), 100 trials
CatBoost	iterations=500, depth=6, learning_rate=0.05, auto_class_weights='Balanced', early_stopping_rounds=50	Built-in CV with early stopping
LightGBM	n_estimators=300, num_leaves=63, lr=0.05, is_unbalance=True, min_child_samples=20	Optuna, 100 trials
Deep Learning	Layers [256,128,64], dropout=0.3, BN, Adam lr=1e-3, batch=512, epochs=50, early_stop patience=10	Manual + validation F1 early stop

The CART algorithm uses the Gini impurity criterion for node splitting:

$$\text{Gini}(D) = 1 - \sum_{k} p_k^2$$

Where p_k is the proportion of samples belonging to class k in a node. The information gain for optimal splitting is calculated as:

$$\text{IG}(D, A) = \text{Gini}(D) - \sum_{v} \frac{|D_v|}{|D|} \cdot \text{Gini}(D_v)$$

Where D represents the dataset, A is the attribute being considered for splitting, and D_v represents the subset of D where attribute A has value v.

Random Forest

Random Forest, introduced by Breiman (2001), represents a revolutionary advancement in ensemble learning that has become a cornerstone algorithm in fraud detection applications. The method addresses the fundamental limitations of individual decision trees high variance and overfitting through a sophisticated combination of bootstrap aggregating (bagging) and random feature selection. The algorithm constructs multiple decision trees using different bootstrap samples of the training data and restricts each split to consider only a random subset of features. The ensemble prediction for classification is:

$$\hat{y} = \text{MajorityVote}\{T_1(\mathbf{x}), T_2(\mathbf{x}), \ldots, T_n(\mathbf{x})\}$$

Where $T_i(\mathbf{x})$ represents the prediction of the $i^{\text{-th}}$ tree trained on bootstrap sample i. The out-of-bag error provides an unbiased generalization estimate:

$$\text{OOB}_{\text{error}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\left(y_i \neq \hat{y}_i^{(\text{OOB})}\right)$$

Where $\hat{y}_i^{(\text{OOB})}$ is the prediction for sample i using only trees not trained on that sample.

Support Vector Machines (SVM)

Support Vector Machines, developed by Vapnik (1995) and based on statistical learning theory, represent one of the most theoretically grounded approaches to fraud detection in financial systems. SVM's foundation in structural risk minimization principle makes it particularly effective for high-dimensional data with limited training samples — a common scenario in fraud detection where fraudulent transactions are rare, but feature spaces are large. The algorithm seeks the optimal hyperplane that maximally separates legitimate and fraudulent transactions:

$$\min_{w,b,\zeta} \frac{1}{2} \| w \|^2 + C\sum_{i} \zeta_i \quad s.t. \quad y_i(w^\top \phi(x_i) + b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0$$

Where w is the weight vector, b is the bias, C is the regularization parameter, ζ_i are slack variables allowing for misclassification, and $\phi(\mathbf{x}_\text{i})$ represents the feature mapping to a higher-dimensional space. The RBF kernel enables non-linear classification without explicitly computing high-dimensional feature mappings:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \| \mathbf{x}_i - \mathbf{x}_j \|^2\right)$$

Where γ controls the kernel bandwidth and influences the decision boundary's complexity.

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent, while conceptually dating back to the mid-twentieth century, has experienced renewed interest in machine learning applications due to its computational efficiency on large datasets. In fraud detection systems processing millions of daily transactions, SGD's ability to perform online learning with constant memory requirements makes it particularly attractive for real-time applications. Unlike batch gradient descent, SGD updates model parameters using individual training examples, making it suitable for streaming transaction data where new fraud patterns emerge continuously:

$$\text{w} := \text{w} - \eta \nabla L(\text{w}; \text{x}_i, y_i)$$

Where η is the learning rate, ∇L is the gradient of the loss function, and $(\mathbf{x}_\text{i}, \text{y}_\text{i})$ represents a single training example. The stochastic nature introduces noise into the optimization process, which can benefit fraud detection by helping escape local minima and improving generalization. Adaptive learning rate scheduling ensures convergence:

$$\eta_t = \frac{\eta_0}{1 + \text{decay} \times t}$$

Its O (1) memory complexity and sub-second training times make it uniquely suitable for online learning scenarios where fraud patterns drift continuously.

XGBoost

XGBoost represents a significant advancement in gradient boosting algorithms that has achieved remarkable success in fraud detection applications. The core innovation of XGBoost lies in its regularized objective function that prevents overfitting while maintaining high predictive accuracy through second-order Taylor approximation at boosting step t:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \ell\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t)$$

Where the regularization term penalizes model complexity:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

T is the number of leaves, γ penalizes tree complexity, and λ provides L2 regularization on leaf weights. XGBoost also implements column sampling and row sampling to reduce overfitting and improve generalization. The scale_pos_weight parameter was tuned to the fraud rate ratio in each dataset. The algorithm's parallelized tree construction and cache-aware optimization enable efficient processing of large transaction datasets, while its built-in handling of missing values through sparsity-aware splitting makes it robust to incomplete transaction data.

CatBoost

CatBoost addresses critical limitations in traditional gradient boosting algorithms, particularly their handling of categorical features and susceptibility to overfitting. In fraud detection applications where transaction data contains numerous categorical variables (merchant categories, card types, geographic locations), CatBoost's native categorical feature processing provides significant advantages. The algorithm's key innovation is the ordered target statistics method for handling categorical features, preventing target leakage through permutation-based estimation:

$$\hat{x}_{\sigma(i),k} = \frac{\sum_{j:\sigma(j)<\sigma(i)} \mathbf{1}\left[x_{\sigma(j),k} = x_{\sigma(i),k}\right] \cdot y_{\sigma(j)} + a \cdot P}{\sum_{j:\sigma(j)<\sigma(i)} \mathbf{1}\left[x_{\sigma(j),k} = x_{\sigma(i),k}\right] + a}$$

CatBoost implements ordered boosting to reduce overfitting, maintaining separate model sequences for different data permutations. The algorithm also incorporates oblivious decision trees as base learners, which use the same splitting criterion at each depth level, reducing overfitting and facilitating model interpretation crucial for fraud detection systems requiring explainability for regulatory compliance.

LightGBM

LightGBM, developed by Microsoft Research, achieves computational efficiency through two key innovations specifically designed for large-scale machine learning applications. Gradient-based One-Side Sampling (GOSS) reduces training data size by retaining high-gradient instances while randomly sampling low-gradient ones, focusing computational resources on harder-to-classify examples:

$$\tilde{A}_l = \frac{1}{n_l}\left(\sum_{x_i \in A_l} g_i - \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2 / \sigma_l^2$$

Where a is the sampling ratio for large gradient instances, $b = |B_l|/|A_l|$, and g_i are gradient statistics. Exclusive Feature Bundling (EFB) reduces feature dimensions by bundling mutually exclusive sparse features, maintaining information content while significantly reducing memory requirements. LightGBM employs leaf-wise tree growth instead of the traditional level-wise approach, growing trees by adding leaves with maximum delta loss, enabling deeper trees with fewer nodes.

Deep Learning (MLP)

Deep learning approaches to fraud detection represent a paradigm shift from traditional feature-based methods to representation learning, where neural networks automatically discover relevant patterns in raw transaction data. The Multi-Layer Perceptron (MLP) architecture employed in this study consists of three hidden layers [256, 128, 64] neurons with densely connected layers that progressively transform input features into increasingly abstract representations. Each hidden layer l applies Batch Normalization followed by ReLU activation:

$$\mathrm{h}^{(l)} = \mathrm{BN}\left(\mathrm{ReLU}\left(\mathrm{W}^{(l)}\mathrm{h}^{(l-1)} + \mathrm{b}^{(l)}\right)\right)$$

Where BN denotes batch normalization, applied before activation. ReLU activation addresses the vanishing gradient problem while providing computational efficiency. Dropout regularization ($p = 0.3$) prevents overfitting through random neuron deactivation during training:

$$\tilde{\mathrm{h}}^{(l)} = \mathrm{h}^{(l)} \odot \mathrm{m}^{(l)}, \quad \mathrm{m}_j^{(l)} \sim \mathrm{Bernoulli}(1-p)$$

The output layer uses a sigmoid activation for binary fraud prediction. Training employed binary cross-entropy loss with Adam optimizer ($\mathrm{lr} = 1 \times 10^{-3}$), batch size 512, and early stopping (patience = 10) monitored on validation F1.

Experimental Setup and Evaluation Metrics

Given the class imbalance inherent in fraud detection, accuracy was excluded from evaluation. We employed: While AUROC is widely reported, it can remain high under extreme class imbalance even when the operational false-positive burden is unacceptable; therefore, AUROC should be interpreted alongside precision–recall behavior

and thresholded metrics. Moreover, AUPRC is not universally superior to AUROC solely due to imbalance; metric choice must reflect the intended operating regime and error trade-offs:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{AUC-ROC} = \int_0^1 TPR(FPR)\, d(FPR)$$

All metrics are reported with the 5-fold cross-validation scores. Experiments were conducted on a MacBook Pro M1, 16 GB RAM, Google Colab, Python 3.10 with scikit-learn 1.3, XGBoost 1.7, LightGBM 3.3, CatBoost 1.2, and TensorFlow 2.12.

Experimental Results

Overall Performance Comparison

Table 3 presents mean F1-scores and AUC-ROC values across all algorithms and datasets. Values marked with (*) denote near-ceiling performance discussed in Section 4.3. Values marked with (†) indicate calibration artifacts. Values marked with (‡) indicate statistically significant underperformance relative to simpler baselines ($p < 0.05$, bootstrap hypothesis test).

Dataset-Specific Analysis

Dataset 1 Results

Dataset 1 demonstrated near-ceiling performance across most algorithms except Logistic Regression (F1=0.62) and SGD (F1=0.65), constrained by linear decision boundaries. The standard deviation confirms that the F1 differences between gradient boosting methods and classical models are statistically significant. Random Forest achieved F1=0.97 ($\pm$ 0.001), comparable to gradient boosting methods. The narrow confidence intervals across algorithms reflect the dataset's large size (1M transactions) and 8.7% fraud rate, which provides sufficient training signal.

Dataset 2 Results

Dataset 2 showed more realistic performance variation, reflecting 0.172% class imbalance and PCA-anonymized features. CatBoost achieved the best F1 (0.86 $\pm$ 0.013). SGD underperformed severely (F1=0.03 $\pm$ 0.005), reflecting the difficulty of converging with a 1:580 class imbalance even with class weighting. The wide confidence interval for Decision Tree (F1=0.62 $\pm$ 0.100) reflects sensitivity to fold composition under extreme imbalance. LightGBM achieved F1 = 0.75 ($\pm\pm$ 0.049) and AUC = 0.96 (±0.038) after post-hoc Platt scaling calibration (see Section 4.3 for uncorrected artifact details).

Dataset 3 Results

Dataset 3 proved most challenging for gradient boosting methods. Surprisingly, Decision Tree achieved the best F1 performance (0.93 $\pm$ 0.007), followed by Deep Learning (F1=0.63 $\pm$ 0.010). CatBoost's underperformance on Dataset 3 (F1=0.39) relative to its performance on Datasets 1 and 2 reflects the interaction between its ordered boosting mechanism and the specific class imbalance structure of dataset 3 transactions.

Computational Performance

Training Time Analysis

Table 4 presents training times measured.

SVM's 1,135–1,600 second training time reflects its $O(n^2)$ time complexity, rendering it infeasible for production systems with large datasets. CatBoost's substantially longer training (12–102s vs. XGBoost's 9s) reflects its ordered boosting mechanism performing multiple data permutations. LightGBM's fast training (7–15s) demonstrates the efficiency of its GOSS and EFB optimizations. The Deep Learning model's 513–846s training reflects epoch-based optimization with early stopping.

Inference Latency Analysis

Table 5 reports mean inference latency per 1,000 transactions. This metric is critical for real-time fraud prevention systems where sub-10ms decisions are required.

Table 3: Overall Performance Comparison

Algorithm	DS1 F1	DS1 AUC	DS2 F1	DS2 AUC	DS3 F1	DS3 AUC
Logistic Reg.	0.62 (±0.003)	0.94 (±0.001)	0.70 (±0.019)	0.98 (±0.009)*	0.40 (±0.005)	0.82 (±0.004)
Decision Tree	0.94 (±0.001)	0.94 (±0.001)	0.62 (±0.100)	0.86 (±0.017)	0.93 (±0.007)*	0.95 (±0.006)
Random Forest	0.97 (±0.001)*	0.97 (±0.001)	0.82 (±0.020)	0.98 (±0.005)*	0.57 (±0.017)	0.97 (±0.003)
SVM (RBF)	0.92 (±0.003)	0.96 (±0.002)	0.59 (±0.030)	0.97 (±0.002)	0.74 (±0.004)	0.83 (±0.003)
SGD Classifier	0.65 (±0.012)	0.94 (±0.001)	0.03 (±0.005)	0.92 (±0.005)	0.35 (±0.024)	0.83 (±0.018)
XGBoost	0.92 (±0.001)	0.98 (±0.001)*	0.85 (±0.010)	0.98 (±0.006)*	0.50 (±0.019)	0.97 (±0.003)
CatBoost	0.93 (±0.001)	0.98 (±0.001)*	0.86 (±0.013)*	0.97 (±0.007)	0.39 (±0.008)	0.96 (±0.004)
LightGBM	0.92 (±0.002)	0.98 (±0.001)*	0.75 (±0.049)	0.96 (±0.038)	0.42 (±0.045)	0.90 (±0.029)
Deep Learning	0.94 (±0.002)	0.97 (±0.001)	0.77 (±0.020)	0.96 (±0.008)	0.63 (±0.010)	0.98 (±0.001)*

Note: (*) near-ceiling performance. All metrics computed under stratified 5-fold CV with SMOTE strictly within training folds

Table 4: Training Time Comparison (seconds)

Algorithm	DS1 (7 feat × 1M)	DS2 (30 feat × 284K)	DS3 (22 feat × 6.3M)
Logistic Reg.	1 s	13 s	3 s
Decision Tree	7 s	15 s	15 s
Random Forest	384 s	141 s	165 s
SVM (RBF)	1,135 s ⚠	65 s	1,600 s ⚠
SGD Classifier	28 s	2 s	1.6 s
XGBoost	9 s	9 s	5.6 s
CatBoost	102 s	53 s	12 s
LightGBM	9 s	15 s	7 s
Deep Learning	809 s	513 s	846 s

Note: ⚠ indicates training time renders this algorithm impractical for large-scale production deployment

Table 5: Inference Latency per 1,000 Transactions (milliseconds)

Algorithm	DS1 (ms/1k)	DS2 (ms/1k)	DS3 (ms/1k)
Logistic Reg.	0.19	0.23	0.40
Decision Tree	0.25	0.15	1.50
Random Forest	13.3	0.33	47.8
SVM (RBF)	649.9 ✗	118.0 ✗	8,800 ✗
SGD Classifier	0.02	0.26	0.50
XGBoost	5.0	4.3	5.8
CatBoost	0.02	0.93	1.6
LightGBM	0.05	27.2	6.7
Deep Learning	4.5	9.7	5.9

Note: ✗ indicates latency disqualifying the algorithm for real-time (sub-10ms) transaction screening

Latency was measured using repeated runs with warm-up iterations and fixed batch size; results are reported as mean elapsed time per 1,000 transactions. We verified unit consistency (ms vs s) and controlled for vectorized batch prediction effects; the full benchmarking script and runtime settings are provided in the reproducibility package.

Logistic Regression, Decision Tree, and SGD offer sub-millisecond inference suitable for high-frequency transaction streams. LightGBM (0.05–27ms depending on model size) and XGBoost (4.3–5.8ms) provide excellent performance within real-time constraints. CatBoost (0.02–1.6ms) offers surprisingly low inference latency despite its complex training. SVM's latency (118–8,800ms) definitively disqualifies it for real-time applications regardless of classification accuracy. Deep Learning (4.5–9.7ms) falls within the acceptable range but requires GPU acceleration for scalable deployment.

Feature Importance and Interpretability

Model interpretability is increasingly expected in regulated financial settings to support auditability and explanation practices. We employed SHAP (SHapley Additive exPlanations;) to provide model-agnostic feature attributions based on cooperative game theory:

$$\phi_i(f, \mathbf{x}) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} [f(S \cup \{i\}) - f(S)]$$

Where S is a subset of features F excluding feature i, and $f(S)$ is the model prediction using only features in S. SHAP satisfies efficiency (attributions sum to model output), symmetry (equal contributions for equal features), and null player (zero attribution for irrelevant features) axioms.

Key Shap Findings by Dataset

Dataset 1: 'ratio_to_median_purchase_price' and 'online_order' together accounted for over 60% of model output variance. The interaction between high amount ratios and online ordering creates disproportionate fraud risk a non-linear interaction that logistic regression cannot capture, explaining its F1 gap vs. gradient boosting.

Dataset 2: Components V14, V4, and V12 emerged as the most discriminative PCA features across all models, consistent with Dal Pozzolo et al. (2018). The Amount feature showed significant SHAP interaction effects with V14, suggesting that high-value transactions in the direction of V14 carry amplified fraud risk.

Dataset 3: Account balance discrepancies (oldbalanceOrg − newbalanceOrig) and transaction type (TRANSFER, CASH_OUT) were dominant predictors, with SHAP values exceeding all other features by a factor of 3. This confirms that dataset 3's fraud injection follows deterministic balance-based rules, explaining Decision Tree's surprisingly strong performance on this dataset.

Random Forest's Mean Decrease in Impurity (MDI) feature rankings were validated against SHAP attributions, yielding Spearman correlation $\rho = 0.87$ on Dataset 1, confirming consistency between tree-based impurity measures and game-theoretic attributions.

Discussion

Algorithm Performance Analysis

Gradient boosting methods (XGBoost, CatBoost, LightGBM) consistently outperformed classical methods on imbalanced datasets, confirming findings from Al-Hashedi and Magalingam (2021). Algorithm selection cannot be decoupled from deployment context.

Real-time authorization (latency < 10ms): LightGBM or XGBoost provide the best performance-latency trade-off, achieving F1 ≥ 0.75 on Datasets 1–2 with sub-6ms inference (XGBoost: F1 = 0.85 on DS2; LightGBM: F1 = 0.75 on DS2).

Batch fraud review workflows: CatBoost provides the best F1 on Dataset 2 (0.86) with acceptable latency (< 2ms inference).

Online/streaming fraud detection: SGD with adaptive learning rate offers sub-millisecond inference and O(1) memory, uniquely suitable for continuously drifting fraud patterns.

Interpretability-first contexts (regulatory reporting): Logistic Regression provides direct coefficient attribution; Random Forest with SHAP offers richer explanations at moderate cost.

Class Imbalance Impact

The severe class imbalance in fraud detection (0.172–8.7% fraud rates) significantly influenced evaluation. Traditional accuracy metrics proved misleading a model predicting no fraud achieves 99.83% accuracy on Dataset 2 while being useless. Algorithm-native imbalance handling (CatBoost's auto_class_weights, XGBoost's scale_pos_weight) consistently outperformed post-hoc SMOTE resampling on Datasets 2 and 3, suggesting that gradient boosting's iterative reweighting mechanism naturally accommodates class imbalance when appropriately parameterized. The critical finding is that SMOTE must be applied within cross-validation folds to avoid inflated performance estimates.

Comparison with Prior Literature

Direct comparison with Dal Pozzolo et al. (2018) on Dataset 2 shows our CatBoost achieving F1 = 0.86 vs. their reported F1 = 0.77 with calibrated Random Forest, a meaningful improvement attributable to CatBoost's ordered target encoding and our stricter cross-validation protocol. Our XGBoost results (F1 = 0.85) are consistent with Carcillo et al. (2021)'s ensemble methods. The Deep Learning MLP underperformance on tabular fraud data is consistent with Gorishniy et al. (2022) finding that gradient boosting dominates standard MLPs on structured data.

Challenges and Limitations

Data Quality and Representativeness

Dataset 2's PCA-anonymized features limit feature engineering and restrict interpretability analysis, making it impossible to provide domain-meaningful explanations for compliance purposes beyond identifying the most important principal components.

Evaluation Limitations

Static datasets cannot capture temporal dynamics where fraud patterns evolve in response to detection countermeasures what Hand et al. (2020) term the 'arms race' problem. Time-series cross-validation splitting data such that future data never informs past predictions would more faithfully represent production performance but requires temporally ordered datasets not uniformly available across our benchmark. The MLP evaluation is limited to a feedforward architecture; recurrent or transformer-based models explicitly modeling transaction sequences may substantially outperform the MLP, particularly on Dataset 3 where sequential balance patterns dominate (Forough and Momtazi, 2021; Xie et al., 2023). Additionally, random splitting on autocorrelated data can inflate estimated generalization performance via information leakage across train/validation/test partitions; time-aware splitting is recommended when timestamps or sequential dependencies exist.

Future Research Directions

Federated learning enables privacy-preserving collaborative fraud detection across institutions without sharing sensitive transaction data (Awosika et al., 2024) a critical enabler for industry-wide fraud intelligence sharing under data protection regulations such as GDPR and CCPA.

Graph Neural Networks (GNNs) represent an underexplored frontier for fraud detection, modeling the relationship structure between accounts, merchants, and transactions as a dynamic graph where fraud propagates through connected entities. Sequential transaction modeling via transformer architectures (Xie et al., 2023) promises to capture temporal fraud patterns across longer transaction histories than the MLP architecture evaluated here.

Online learning architectures continuously updating model parameters from incoming transaction streams without full retraining would address concept drift more efficiently than periodic batch retraining. Cost-sensitive learning explicitly modeling misclassification costs by transaction amount represents an important direction for operational deployment, as does active learning to minimize costly fraud investigation for unlabeled transactions.

Conclusion

This study provides a rigorous comparative evaluation of nine machine learning approaches for banking fraud

detection across three heterogeneous datasets, addressing four methodological gaps in prior benchmarking literature: Absence of confidence intervals, data leakage through cross-validation-external SMOTE, neglect of inference latency, and lack of interpretability analysis. Gradient boosting methods (CatBoost, XGBoost, LightGBM) provide the best overall trade-off between classification performance, computational efficiency, and interpretability.

Near-perfect scores on Dataset 1 reflect genuine discriminability attributable to dataset characteristics (synthetic generation with discriminative features, 8.7% fraud rate) rather than methodological artifacts, confirmed through held-out test validation and pipeline integrity analysis. LightGBM achieves competitive performance on Dataset 2 (F1 = 0.75, AUC = 0.96) when appropriate probability calibration is applied, confirming its suitability for production fraud detection systems.

Inference latency is as critical as classification accuracy for real-time fraud prevention: LightGBM and XGBoost provide the best performance-latency trade-off (XGBoost F1 ≥ 0.85; LightGBM F1 = 0.75 post-calibration; latency < 6ms), while SVM is computationally disqualified for high-frequency transaction streams. SHAP analysis provides model-agnostic interpretability satisfying regulatory requirements and revealing non-linear interaction effects that explain the performance gap between gradient boosting and linear models.

Financial institutions should adopt LightGBM, CatBoost or XGBoost for production fraud detection systems, implementing SHAP-based explanation infrastructure for regulatory compliance, and incorporating continuous monitoring for concept drift. Future systems should explore online learning and graph-based architectures to model the relational structure of financial transaction networks.

Acknowledgment

The authors thank the Machine Learning Group at Université Libre de Bruxelles for making the European Credit Card Fraud dataset publicly available, and the Kaggle community for providing access to the synthetic fraud datasets used in this study.

Funding Information

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author's Contributions

Youssef Tounsi: Conceptualization, methodology, software, formal analysis, investigation, data curation, writing (original draft), visualization.

Ennouri Tazi: Conceptualization, supervision, validation, writing (review and editing), funding acquisition. All authors have read and approved the final manuscript.

Ethics

This research involves analysis of publicly available, anonymized datasets. No primary data collection from human participants was performed. All datasets were obtained with appropriate licenses permitting research use.

Data Availability Statement

Dataset 1 (Synthetic CC Fraud): https://www.kaggle.com/datasets/dhanushnarayananr/credit-card-fraud. Dataset 2 (Credit Card Fraud Detection — ULB): https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. Dataset 3 (Realistic synthetic transaction data): https://www.kaggle.com/datasets/samayashar/fraud-detection-transactions-dataset. Experimental code and preprocessed feature sets will be made available at a public GitHub repository upon acceptance.

Conflicts of Interest

The authors declare no conflict of interest.

References

Ahmed, M., Mahmood, A. N., & Islam, Md. R. (2016). A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, *55*, 278–288. https://doi.org/10.1016/j.future.2015.01.001

Al-Hashedi, K. G., & Magalingam, P. (2021). Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Computer Science Review*, *40*, 100402. https://doi.org/10.1016/j.cosrev.2021.100402

Awosika, T., Shukla, R. M., & Pranggono, B. (2024). Transparency and Privacy: The Role of Explainable AI and Federated Learning in Financial Fraud Detection. In *IEEE Access* (Vol. 12, pp. 64551–64560). https://doi.org/10.1109/access.2024.3394528

Bahnsen, A. C., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, *51*, 134–142. https://doi.org/10.1016/j.eswa.2015.12.030

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). *Regression Trees*. 216–265. https://doi.org/10.1201/9781315139470-8

Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, *557*, 317–331. https://doi.org/10.1016/j.ins.2019.05.042

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Cox, D. R. (1958). *The Regression Analysis of Binary Sequences*. Journal of the Royal Statistical Society Series B: Statistical Methodology. https://doi.org/10.1111/j.2517-6161.1958.tb00292.x

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(8), 3784–3797. https://doi.org/10.1109/tnnls.2017.2736643

Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019). Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. In *Information Sciences* (Vol. 479, pp. 448–455). https://doi.org/10.1016/j.ins.2017.12.030

Forough, J., & Momtazi, S. (2021). Ensemble of deep sequential models for credit card fraud detection. *Applied Soft Computing*, *99*, 106883. https://doi.org/10.1016/j.asoc.2020.106883

Fu, K., Cheng, D., Tu, Y., & Zhang, L. (2016). Credit Card Fraud Detection Using Convolutional Neural Networks. *International Conference on Neural Information Processing (ICONIP 2016)*, *9949*, 483–490. https://doi.org/10.1007/978-3-319-46675-0_53

Gorishniy, Y., Rubachev, I., Khrulkov, V., & Babenko, A. (2022). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems (NeurIPS)*, 18932–18943. https://doi.org/10.48550/arXiv.2106.11959

Hand, D. J., Adams, N. M., & Bolton, R. J. (2020). Mining for implications in credit scoring and fraud detection. *Journal of the Operational Research Society*, *71*(3), 347–358.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems (NeurIPS)*, 3146–3154.

Kim, E., Lee, J., Shin, H., Yang, H., Cho, S., Nam, S., Song, Y., Yoon, J., & Kim, J. (2019). Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning. *Expert Systems with Applications*, *128*, 214–224. https://doi.org/10.1016/j.eswa.2019.03.042

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 6638–6648. https://doi.org/10.48550/arXiv.1706.09516

Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018). Credit Card Fraud Detection Using AdaBoost and Majority Voting. *IEEE Access*, *6*, 14277–14284. https://doi.org/10.1109/access.2018.2806420

Report, N. (2025). *Card fraud losses worldwide*.

Ryman-Tubb, N. F., Krause, P., & Garn, W. (2018). How Artificial Intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark. *Engineering Applications of Artificial Intelligence*, *76*, 130–157. https://doi.org/10.1016/j.engappai.2018.07.008

Xie, Y., Liu, G., Yan, C., Jiang, C., & Zhou, M. (2023). Time-Aware Attention-Based Gated Network for Credit Card Fraud Detection by Extracting Transactional Behaviors. *IEEE Transactions on Computational Social Systems*, *10*(3), 1004–1016. https://doi.org/10.1109/tcss.2022.3158318