Journal of Data Science and Modeling, Vol. 4, No. 1, 1-30, December 2025

Research Manuscript

A System-Oriented Framework for Credit Card Fraud Detection Using Feature Engineering and Threshold Optimization

Negin Kaveh[1], Vadood Karamati[1]

[1]Faculty of Computer Science, Allameh Tabataba'i University, Tehran, Iran

Recieved: 08/06/2026 Accepted: 06/11/2026

Abstract: This study presents a credit card fraud detection framework integrating feature engineering, optional SMOTE, XGBoost, MCC–cost threshold optimization, and SHAP explainability. After duplicate removal, European transaction data were split into stratified training, validation, and test sets. Data-dependent preprocessing and SMOTE used training data only; threshold selection used validation data. Independent test performance reached ROC-AUC 0.9698, PR-AUC 0.8136, and MCC 0.8319, with a classification cost of 228. Evaluation included bootstrap intervals, McNemar testing, ablation, cost sensitivity, and PSI analysis. Operational outputs are APPROVE, REVIEW, and BLOCK. A 1,000-sample batch benchmark supports deployment feasibility, subject to infrastructure-dependent production latency. Low PSI indicates train–test distributional similarity; long-term temporal stability and adaptation remain untested.

Keywords: Credit Card Fraud Detection, Imbalanced Data, Threshold Optimization, Matthews Correlation Coefficient (MCC), XGBoost
Mathematics Subject Classification (2010): 68T05.

1. Introduction

Credit card fraud has become one of the most significant threats to financial security in the digital economy. According to international reports, billions of dollars are lost annually due to fraudulent payment activities. The rapid growth of online transactions and the increasing sophistication of fraud strategies have made the application of advanced machine learning techniques essential.

One of the main challenges in this domain is the severe class imbalance in financial datasets. In most real-world fraud datasets, fraudulent transactions represent less than 1% of the total observations. This imbalance makes traditional performance measures such as Accuracy misleading, since models may achieve high accuracy simply by predicting the majority class.

In many previous studies, the primary focus has been on selecting or comparing classification algorithms. However, in real-world banking environments, an effective fraud detection system must incorporate several components, including:

- Data management
- Feature extraction
- Imbalance handling
- Decision threshold determination
- Risk analysis
- Operational decision generation

The main distinction of this study is that fraud detection is addressed as a complete decision-making system rather than merely a classification task.

The main objective of this research is to design an operational fraud detection framework that integrates data preprocessing, imbalance handling, model training, and decision optimization into a unified system architecture. Unlike many previous studies that only compare classification algorithms, this study focuses on building a practical decision-support system capable of generating operational decisions for real financial environments. The experimental results demonstrate that the proposed framework achieves high detection performance while maintaining system stability and interpretability. The main contributions of this study are as follows:

1. A system-oriented fraud detection framework that integrates data preprocessing, feature engineering, SMOTE-based imbalance handling, threshold optimization, and post-hoc explainability within a unified and reproducible pipeline.

2. A composite objective for threshold selection, combining MCC maximization and classification cost minimization. Both components are explicitly normalized and aligned in the same direction to form a single composite objective function.

3. A leakage-free experimental design in which, after dataset-level duplicate removal, the train/validation/test split (stratified by target class) is performed before fitting any data-dependent transformation, the StandardScaler is fitted only on the training set, and SMOTE is applied exclusively to the training set. Threshold optimization and all reported metrics are computed on untouched validation and test sets.

4. A comprehensive evaluation protocol including ROC-AUC, PR-AUC, MCC, precision, recall, F1-score, Brier score, bootstrap confidence intervals, McNemar testing, ablation analysis, cost sensitivity analysis, and SHAP-based interpretability, providing a multi-perspective assessment of the framework.

5. Distribution drift analysis based on the Population Stability Index (PSI) to verify the distributional similarity between training and test partitions in the present dataset, rather than asserting long-term concept drift.

6. An operational decision engine that maps predicted probabilities into three actionable categories (APPROVE, REVIEW, BLOCK), enabling the framework to be directly embedded into fraud risk management workflows.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the conceptual model. Section 4 describes the methodology. Section 5 reports experimental results. Section 6 discusses practical implications and limitations. Section 7 concludes the paper, and Section 8 outlines future work.

2. Related Work

2.1 Traditional Approaches to Fraud Detection

Early fraud detection systems primarily relied on rule-based approaches and expert systems. These methods used predefined rules to flag suspicious transactions based on thresholds and patterns. While simple to implement, these systems struggled to adapt to new fraud patterns and generated high false positive rates.

2.2 Machine Learning-Based Fraud Detection

With the advancement of machine learning, researchers began applying various classification algorithms to fraud detection. Studies have explored Logistic Regression, Decision Trees, Random Forest, Support Vector Machines, and Neural Networks. Dal Pozzolo et al. (2015) investigated the calibration of probability estimates in fraud detection using undersampling techniques.

2.3 Ensemble Methods and Deep Learning

Ensemble methods, particularly Random Forest (Breiman , 2001) and Gradient Boosting, have shown promising results in fraud detection. Chen and Guestrin (2016) introduced XGBoost, which has become widely adopted due to its scalability and performance. Liu et al. (2024) proposed a hybrid CNN-SVM-LightGBM model for enhanced credit card fraud detection on imbalanced datasets, demonstrating the effectiveness of combining multiple models.

2.4 Handling Class Imbalance

The class imbalance problem has been addressed through various techniques, including undersampling, oversampling, and hybrid approaches. Chawla et al. (2002) introduced SMOTE (Synthetic Minority Over-sampling Technique), which generates synthetic samples for the minority class. This technique has been widely adopted in fraud detection research.

2.5 Model Interpretability and Explainability

Recent research has emphasized the importance of model interpretability in financial applications. Hasan et al. (2024) compared explainable AI techniques for anomaly detection in financial transactions. Explainability is crucial for building trust in automated decision-making systems and meeting regulatory requirements.

2.6 Systematic Comparative Analysis

Beyond the categories presented above, a systematic examination of existing literature reveals that most studies have focused primarily on algorithm-level performance comparison, with less emphasis on designing a comprehensive decision-support architecture. Many studies rely solely on ROC-AUC and do not incorporate metrics suitable for highly imbalanced data, such as MCC.
Furthermore, in most studies:

- The decision threshold is fixed at the default value of 0.5

- No cost-sensitive error analysis is performed
- Model drift and long-term stability are not evaluated
- Operational deployment architectures are not discussed

These limitations highlight the research gap in developing fully structured fraud detection systems for industry.

2.7 Research Gap

Most previous studies:

- Have compared only a single model
- Have relied on the default threshold of 0.5
- Have not performed stability or drift analysis
- Have not generated operational decision outputs

This study addresses these gaps by designing a complete and multi-stage fraud detection system. Ullah et al. (2022) introduced an adaptive system for big data cybersecurity analytics. Moder et al. (2026) investigated decision threshold setting in binary classification problems from a behavioral perspective. Baesens et al. (2021) emphasized the need for data engineering approaches in fraud detection. However, the majority of these studies remain focused on algorithmic aspects and do not provide a unified, deployment-ready pipeline that jointly addresses imbalance handling, threshold optimization, cost sensitivity, distribution drift, and explainability.

Table 1 summarizes the comparison between selected existing studies and the proposed framework.

3. Conceptual Model of the System

The proposed fraud detection system is designed as a multistage intelligent pipeline that processes transaction data through several analytical layers. The system architecture includes data preprocessing, feature engineering, class imbalance handling, machine learning model training, decision threshold optimization, and operational decision generation.

Figure 1 illustrates the overall architecture of the proposed system.

The conceptual model demonstrates that the proposed system consists of a complete pipeline, starting from raw transaction data and ending with an operational decision. The flow is as follows:

Table 1: Comparative Analysis of Existing Fraud Detection Studies and the Proposed Framework

Criterion	LR + Under.	GAN	Ensemble	Proposed
Threshold tuning	No	No	No	Yes
MCC	No	No	Limited	Yes
Explainability	No	No	No	SHAP
Drift	No	No	Partial	Yes
Deployment	No	No	No	Yes

Note: LR + Under. = Logistic Regression with undersampling (Dal Pozzolo et al. , 2015); GAN (Fiore et al. , 2019); Ensemble (Carcillo et al. , 2021); Proposed = XGBoost + SMOTE.

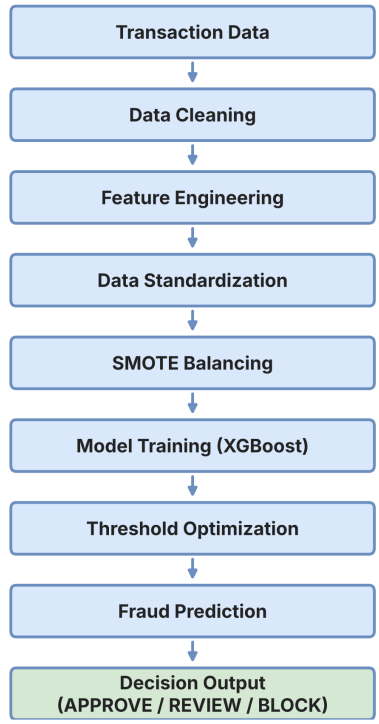


Figure 1: Conceptual architecture of the proposed credit card fraud detection system

Transaction Data → Data Cleaning → Feature Engineering → Data Standardization → SMOTE Balancing → Model Training → Threshold Optimization → Fraud Prediction → Decision Output (BLOCK / REVIEW / APPROVE)

4. Research Methodology

The proposed fraud detection framework is designed as a multi-stage intelligent system that processes transaction data through several analytical layers. Each stage of the system is designed to improve the reliability and interpretability of fraud detection while maintaining scalability for real-world financial environments.

4.1 System Design Overview

The proposed framework is designed as an intelligent, system-oriented credit card fraud detection pipeline. The main goal of the system is not only to classify transactions as fraudulent or legitimate, but also to provide a practical decision-support structure that can be directly used in financial environments. To achieve this, the framework combines preprocessing, feature engineering, imbalance correction, supervised learning, threshold tuning, and operational decision generation in a unified architecture.

4.2 Data Preprocessing and Cleaning

The raw transaction data are first cleaned to remove inconsistencies, missing values, and redundant records. The raw transaction data contained 1,081 duplicate records, which were removed during the preprocessing phase. After removing duplicates, the final dataset consisted of 283,726 transactions. Since financial transaction datasets are usually noisy and highly imbalanced, preprocessing is a necessary step to improve the quality of the input data before model training.
Key steps in data preprocessing:

- Removal of duplicate records
- Handling missing values
- Data type validation

After cleaning, feature engineering is performed to improve the representation of transaction behavior. Informative variables are retained and transformed where necessary.

4.3 Feature Engineering

Two additional features were created to improve the representation of transaction behavior:

- **LogAmount:** Logarithmic transformation of transaction amount to handle its strong right-skewness, defined by

$$\text{LogAmount} = \log(1 + \text{Amount}).$$

- **Hour:** Extracted from the original Time column, which records the seconds elapsed since the first transaction in the dataset. The hourly index was computed as

$$\text{Hour} = \lfloor \text{Time}/3600 \rfloor \bmod 24,$$

  mapping each transaction to a 0–23 hourly index. This captures within-day temporal patterns in fraudulent activity.

4.4 Data Standardization

Standardization is applied to ensure that all features are on a comparable scale, which helps improve the stability and convergence of machine learning algorithms. The StandardScaler is fitted only on the training data to prevent data leakage.

4.5 Handling Class Imbalance with SMOTE

Because fraudulent transactions represent only a very small portion of the dataset (0.167%), class imbalance is a major challenge. To address this issue, the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al. , 2002) is used. SMOTE generates synthetic minority samples to balance the class distribution and reduce the bias of the classifier toward the majority class.
SMOTE Formula:

$$x_{\text{new}} = x_i + \lambda\,(x_{\text{nn}} - x_i) \tag{4.1}$$

where x_i is a minority sample, x_{nn} is its nearest neighbor, and λ is a random number between 0 and 1.

Preprocessing sequence (leakage prevention): After dataset-level duplicate removal, the data were divided into training, validation, and test sets using stratified sampling (to preserve class ratios) before fitting any data-dependent transformation or applying SMOTE. The StandardScaler was fitted only on the training set and subsequently applied to the validation and test sets. Likewise, SMOTE was applied exclusively to the training set after the split; no synthetic

samples were generated for the validation or test sets. Threshold optimization and all evaluation metrics were computed on the untouched validation and test sets, respectively. This procedure ensures that the reported test performance reflects the model's behavior on genuinely unseen data.

4.6 Model Training and Hyperparameters

A supervised machine learning model is trained on the processed dataset to distinguish between fraudulent and non-fraudulent transactions.
Model: XGBoost Classifier (Chen and Guestrin , 2016)
Hyperparameters:

- `n_estimators`: 500
- `max_depth`: 6
- `learning_rate`: 0.05
- `subsample`: 0.8
- `colsample_bytree`: 0.8
- `eval_metric`: logloss
- `random_state`: 42

4.7 Decision Threshold Optimization

Instead of using the default classification threshold of 0.50, the decision threshold is optimized to improve the trade-off between precision and recall. This step is especially important in fraud detection, where missing a fraudulent transaction can be more costly than incorrectly flagging a legitimate one.
Optimization Objective:

$$\text{Objective} = 0.7\,\text{MCC}_{\text{Normalized}} + 0.3\,\text{Cost}_{\text{Normalized}} \tag{4.2}$$

Since MCC is intended to be maximized while classification cost is intended to be minimized, both components are first normalized to the $[0, 1]$ interval and aligned in the same direction:

$$\text{MCC}_{\text{norm}}(t) = \frac{\text{MCC}(t) - \min \text{MCC}}{\max \text{MCC} - \min \text{MCC}} \tag{4.3}$$

$$\text{Cost}_{\text{norm}}(t) = 1 - \frac{\text{Cost}(t) - \min \text{Cost}}{\max \text{Cost} - \min \text{Cost}} \tag{4.4}$$

where $\text{Cost}(t) = C_{\text{FP}}\,\text{FP}(t) + C_{\text{FN}}\,\text{FN}(t)$, with $C_{\text{FP}} = 1$ and $C_{\text{FN}} = 10$. The final composite objective function is:

$$J(t) = 0.70\,\text{MCC}_{\text{norm}}(t) + 0.30\,\text{Cost}_{\text{norm}}(t) \tag{4.5}$$

The threshold $t^* = \arg\max_t J(t)$ is selected on the validation set and applied to the test set.

Decision Rules:

- Probability $\geq$ Threshold (0.6600) $\rightarrow$ BLOCK
- Probability $\leq$ Review Threshold (0.4620) $\rightarrow$ APPROVE
- Review Threshold (0.4620) $\leq$ Probability $\leq$ Threshold (0.6600) $\rightarrow$ REVIEW

The review threshold is heuristically defined as 70% of the optimal decision threshold ($0.6600 \times 0.70 = 0.4620$). It is not separately optimized or learned from data. This buffer zone allows for manual inspection of uncertain predictions. While this review category is intended to support human oversight of borderline cases, its operational effectiveness in reducing false positives and false negatives requires separate empirical evaluation and remains outside the scope of the present study.

4.8 Evaluation Metrics

Evaluation metrics used in this study include (Ferri et al. , 2009):

- **ROC-AUC:** Area Under the Receiver Operating Characteristic Curve
- **PR-AUC:** Area Under the Precision–Recall Curve (more suitable for imbalanced data)
- **MCC (Matthews Correlation Coefficient)** (Powers , 2011)**:**

$$\text{MCC} = \frac{TP\,TN - FP\,FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{4.6}$$

- **Precision:** $\text{Precision} = TP/(TP+FP)$
- **Recall:** $\text{Recall} = TP/(TP+FN)$
- **F1-score:** $F_1 = 2\,\text{Precision}\,\text{Recall}/(\text{Precision} + \text{Recall})$
- **Cost function:** $\text{CostFunction} = FN\,C_{\text{FN}} + FP\,C_{\text{FP}}$

4.9 Statistical Significance Testing

To ensure the reliability of results, statistical significance tests are performed. McNemar's test is conducted using the exact binomial version (no continuity correction) to compare the proposed XGBoost model with a Logistic Regression baseline.

4.10 System Stability and Drift Analysis

To assess reliability, the framework includes:

- **Ablation Study:** Systematic removal of components to evaluate their contribution.
- **Drift Analysis (PSI):**

$$\text{PSI} = \sum_i (\text{Actual}_i - \text{Expected}_i) \ln(\text{Actual}_i / \text{Expected}_i) \tag{4.7}$$

4.11 Model Interpretability with SHAP

To improve interpretability, SHAP (SHapley Additive exPlanations) analysis is performed. SHAP values quantify the contribution of each feature to individual prediction outcomes and provide transparent explanations for model decisions.

4.12 Experiment Configuration

To ensure reproducibility, all experiments were conducted with the following fixed configuration:

- Random State: 42
- Test Size: 20% of the total data
- Validation Size: 20% of the total dataset (implemented as 25% of the training partition after the test split)
- Split Strategy: Stratified by target class
- SMOTE Random State: 42
- Threshold Search Range: 0.05 to 0.95 with step size 0.01
- False Positive Cost: 1
- False Negative Cost: 10
- MCC Weight in Objective Function: 0.70
- Cost Weight in Objective Function: 0.30

4.13 Software and Libraries

The proposed system was implemented using Python 3.10.9 with the following key libraries:

- NumPy 1.24.4
- Pandas 2.3.3
- Scikit-learn 1.3.2
- XGBoost 1.7.6
- Imbalanced-learn 0.11.0
- SHAP 0.44.1
- Statsmodels 0.14.6

5. Results

5.1 Dataset Description

The dataset used in this study is the widely known Credit Card Fraud Detection Dataset, publicly available on Kaggle. The dataset contains real credit card transactions made by European cardholders over two consecutive days in 2013.
Key characteristics:

- The data are anonymized
- Variables V1 to V28 are the result of PCA transformation
- Additional features: Amount and Time
- Target variable: Class (Fraud = 1, Normal = 0)

Table 2 summarizes the characteristics of the dataset.

Table 2: Dataset Characteristics

Description	Value
Number of Transactions (Rows)	283,726
Number of Features (Columns)	33
Target Variable	Class (Fraud/Normal)

Summary statistics of the dataset are given in Table 3.

Table 3: Dataset Summary Statistics

Metric	Value
Total Rows	283,726
Fraud Transactions	473
Normal Transactions	283,253
Fraud Ratio	0.167%
Number of Features	32

5.2 Data Split

Table 4 shows the resulting stratified data split.

Table 4: Data Split Summary

Dataset	Rows	Percentage	Fraud	Fraud Ratio
Training	170,235	60%	284	0.1668%
Validation	56,745	20%	94	0.1657%
Testing	56,746	20%	95	0.1674%

Note: The validation set represents 20% of the full dataset and 25% of the training partition after the test split.

5.3 Threshold Optimization Results

The decision threshold was optimized on the validation set using a composite objective function that balances the normalized MCC and normalized cost. The search was performed over the range of 0.05 to 0.95 with a step size of 0.01.

Table 5 reports the optimal decision threshold and the corresponding validation metrics.

Table 5: Optimal Decision Threshold Based on MCC and Cost Optimization

Metric	Value
Best Threshold	0.6600
Validation MCC	0.8804
Validation Cost	139.0

5.4 Classification Performance

Table 6 presents the per-class classification performance on the test set.

Table 6: Classification Performance Metrics

Class	Precision	Recall	F1-score	Support
Normal (0)	1.00	1.00	1.00	56,651
Fraud (1)	0.90	0.77	0.83	95
Accuracy	—	—	0.9995	56,746
Macro avg	0.95	0.88	0.91	56,746
Weighted avg	1.00	1.00	1.00	56,746

5.5 Confusion Matrix

Table 7 presents the confusion matrix on the test set.

Table 7: Confusion Matrix

	Predicted Normal	Predicted Fraud
Actual Normal	56,643	8
Actual Fraud	22	73

Figure 2 visualizes the confusion matrix reported in the table above.

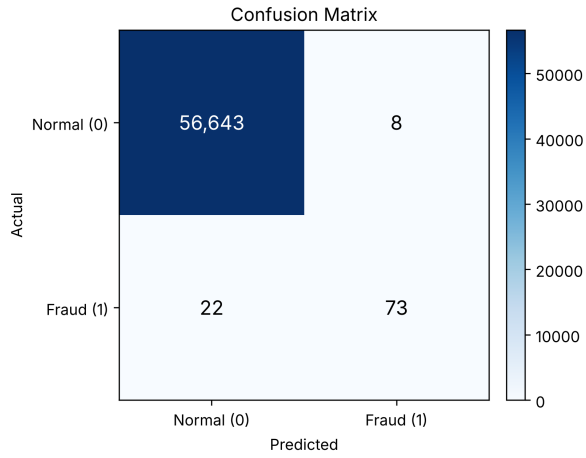


Figure 2: Confusion Matrix

5.6 Final Metrics Summary

The final performance of the proposed system on the independent test set is summarized in Table 8. The model achieves a strong ROC-AUC of 0.9698 and a

PR-AUC of 0.8136, demonstrating excellent discriminative ability even under severe class imbalance. The MCC of 0.8319 confirms balanced performance across both classes.

Table 8: Final Performance Metrics

Metric	Value
ROC-AUC	0.9698
PR-AUC	0.8136
MCC	0.8319
Precision (Fraud)	0.9012
Recall (Fraud)	0.7684
F1-Score (Fraud)	0.8295
Accuracy	0.9995
FPR	0.00014
FNR	0.2316
Brier Score	0.0005
Fraud Detected	73
Fraud Missed	22
False Positives	8
Total Cost	228

5.7 ROC and PR Curve

Figure 3 shows the ROC curve, and Figure 4 shows the Precision–Recall curve of the proposed model on the test set.

5.8 Model Comparison

Table 9 compares the proposed XGBoost model with the Logistic Regression and Random Forest baselines.

Figure 5 compares the MCC of the three models.

5.9 Feature Importance

Table 10 lists the ten most important features according to the XGBoost model.

Figure 6 displays the importance ranking of the top 10 features.

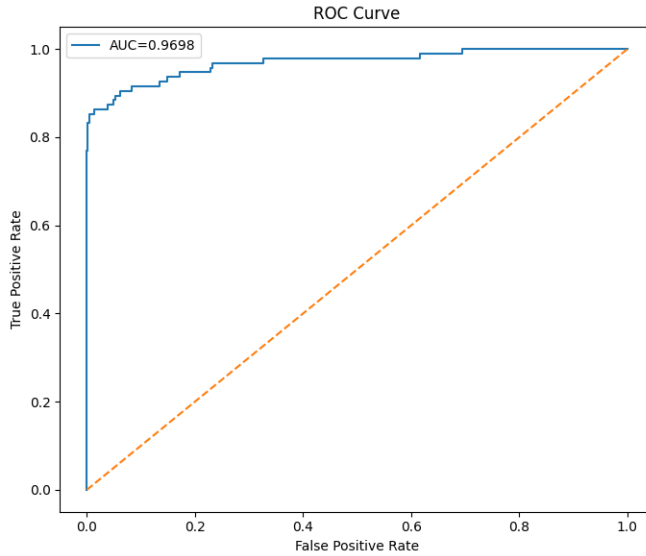


Figure 3: ROC Curve for Fraud Detection Model

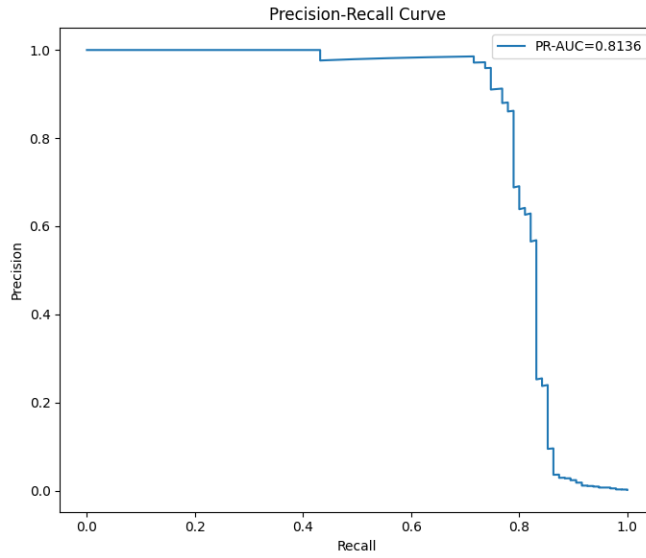


Figure 4: Precision–Recall Curve for Fraud Detection Model

Table 9: Model Comparison

Metric	Logistic Regression	Random Forest	XGBoost (Proposed)
Threshold	0.95	0.49	0.66
MCC	0.4482	0.8182	0.8319
ROC-AUC	0.9619	0.9444	0.9698
PR-AUC	0.6886	0.8155	0.8136
Precision	0.2431	0.9091	0.9012
Recall	0.8316	0.7368	0.7684
F1	0.3762	0.8140	0.8295

Note: The thresholds reported for Logistic Regression, Random Forest, and XGBoost in this table were selected by maximizing MCC alone on the same validation set. The operational threshold of the proposed framework (0.66) was selected using the composite MCC–cost objective. This difference in threshold-selection criteria should be considered when comparing threshold values across models.

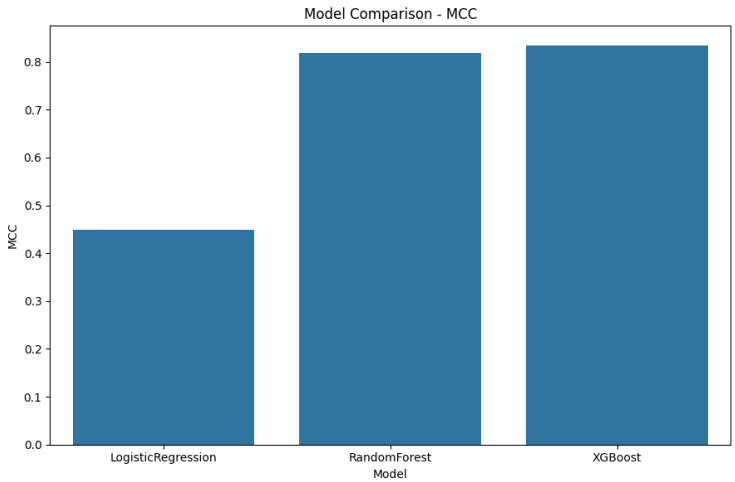


Figure 5: Performance Comparison of Logistic Regression, Random Forest, and XGBoost Classifier

Table 10: Top 10 Features by Importance

Rank	Feature	Importance
1	V14	0.4412
2	V10	0.0978
3	V4	0.0715
4	V12	0.0674
5	V8	0.0420
6	V17	0.0400
7	V20	0.0144
8	V11	0.0142
9	Amount	0.0137
10	V18	0.0130

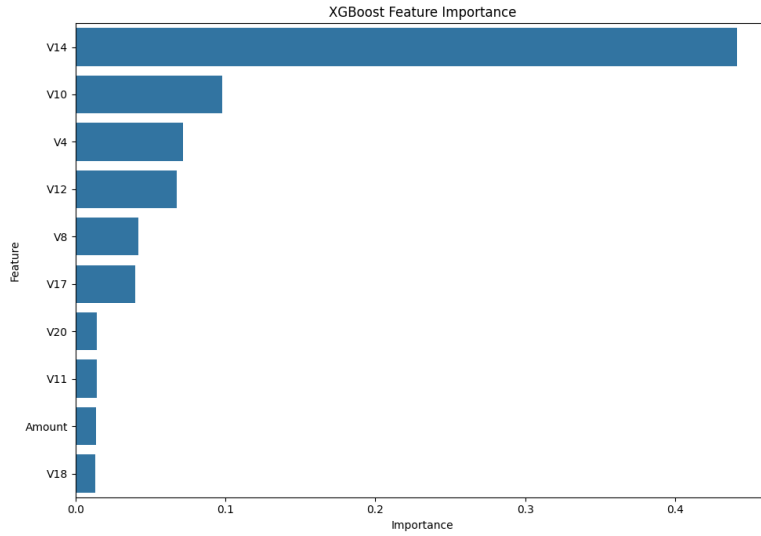


Figure 6: Feature importance ranking of the top 10 predictors in the XGBoost model

5.10 Bootstrap Confidence Intervals

Table 11 reports bootstrap confidence intervals for ROC-AUC, PR-AUC, and MCC.

Table 11: Bootstrap Confidence Intervals (1000 iterations)

Metric	Mean	Std	CI 2.5%	CI 97.5%
ROC-AUC	0.9699	0.0115	0.9445	0.9887
PR-AUC	0.8140	0.0378	0.7357	0.8858
MCC	0.8314	0.0291	0.7689	0.8846

McNemar's test was conducted using the exact binomial version (no continuity correction) to compare the proposed XGBoost model with the Logistic Regression baseline on the test set. The test yielded a p-value less than 0.001, indicating a statistically significant difference in classification performance between the two models ($p < 0.05$).

5.11 Ablation Study

Table 12 reports the ablation results.

Table 12: Ablation Study Results

Metric	Full	No SMOTE	No Hour	No LogAmt	No Both
Threshold	0.94	0.11	0.80	0.92	0.92
MCC	0.8336	0.8444	0.8382	0.8399	0.8236
ROC-AUC	0.9698	0.9726	0.9711	0.9689	0.9684
PR-AUC	0.8136	0.8194	0.8238	0.8094	0.8135
Precision	0.9714	0.9036	0.9024	0.9718	0.9211
Recall	0.7158	0.7895	0.7789	0.7263	0.7368
F1	0.8242	0.8427	0.8362	0.8313	0.8187

Note: Full = Full Model; No LogAmt = Without LogAmount; No Both = Without Hour + LogAmount. Ablation thresholds were selected by maximizing MCC alone on the validation set, whereas the final operational threshold of the proposed framework was selected using the composite MCC-cost objective.

It should be noted that the ablation experiments and the final proposed framework use different threshold-selection criteria. In the ablation study, the optimal threshold (e.g., 0.94 for the Full Model) was selected by maximizing MCC alone on the validation set. In contrast, the final framework adopts the composite MCC-cost objective described in Section 4.7, which yields an operational threshold of 0.66. Consequently, the difference between the reported thresholds reflects a change in

the selection criterion rather than a difference in the underlying model or training procedure.

Regarding the role of SMOTE, the ablation results show that the model trained without SMOTE achieved marginally higher MCC (0.8444 vs. 0.8336) and F1-score (0.8427 vs. 0.8242) on this specific dataset. However, SMOTE remains an optional imbalance-handling component of the proposed framework, particularly for scenarios in which the minority class is extremely scarce. Its effect on generalization may vary across datasets, cost structures, and evaluation criteria; therefore, SMOTE should be regarded as an important building block of the imbalance-handling pipeline rather than a guaranteed performance enhancer in every setting.

Figure 7 summarizes the ablation results graphically.

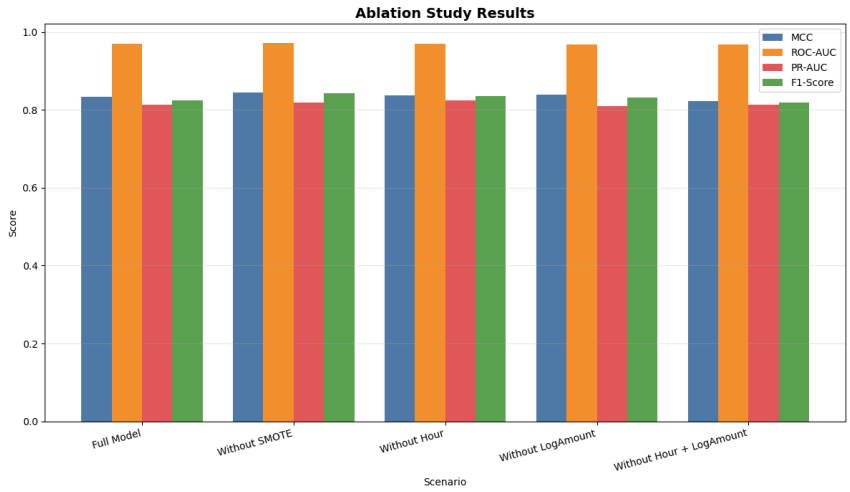


Figure 7: Ablation Study Results for the Proposed Fraud Detection Framework

5.12 Drift Analysis

Table 13 reports the PSI-based drift scores.

Table 13: Drift Analysis Results

Drift Type	PSI Score
Time Distribution Drift	0.00031
Maximum Feature Distribution Drift	0.00050
Mean Feature Distribution Drift	0.00023

Figure 8 presents the PSI scores reported above.

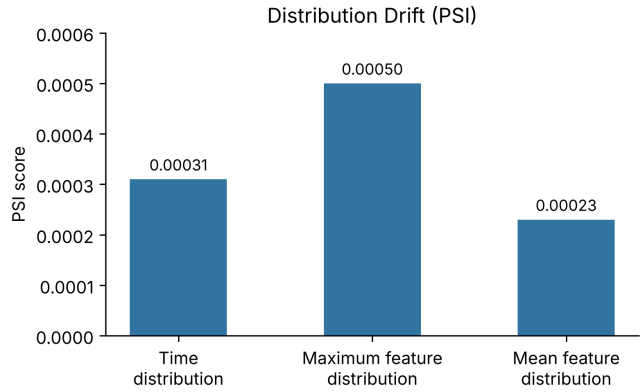


Figure 8: Distribution Drift Analysis Based on the Population Stability Index (PSI)

The PSI-based analysis reported in this study evaluates changes in the distributions of features and time between the training and test sets and is therefore referred to as distribution drift (data drift) analysis. All reported PSI values are well below the conventional threshold of 0.1 (maximum feature PSI = 0.00050, mean feature PSI = 0.00023, time PSI = 0.00031), indicating that the feature and time distributions are highly stable between the training and test partitions. The low PSI values indicate distributional similarity between the training and test partitions in the present dataset. Long-term temporal stability and adaptation to evolving fraud patterns were not evaluated and remain subjects for future research.

5.13 SHAP Feature Importance

Table 14 lists the 15 features with the highest mean absolute SHAP values.

Figure 9 shows the SHAP summary plot.

5.14 Cost Sensitivity Analysis

Table 15 presents the cost sensitivity analysis for different cost settings.

5.15 System Performance

Table 16 reports system performance from the batch benchmark.

The model was benchmarked using a batch-inference benchmark of 1,000 test samples on a system with an Intel Core i7-10750H processor and 16 GB of RAM.

Table 14: Top 15 Features by Mean Absolute SHAP Value

Rank	Feature	Mean |SHAP|
1	V14	2.335
2	V4	1.661
3	Hour	1.028
4	V8	0.882
5	V11	0.805
6	LogAmount	0.675
7	V3	0.637
8	Amount	0.581
9	V10	0.578
10	V12	0.518
11	V15	0.474
12	V1	0.412
13	V5	0.404
14	V17	0.364
15	V26	0.359

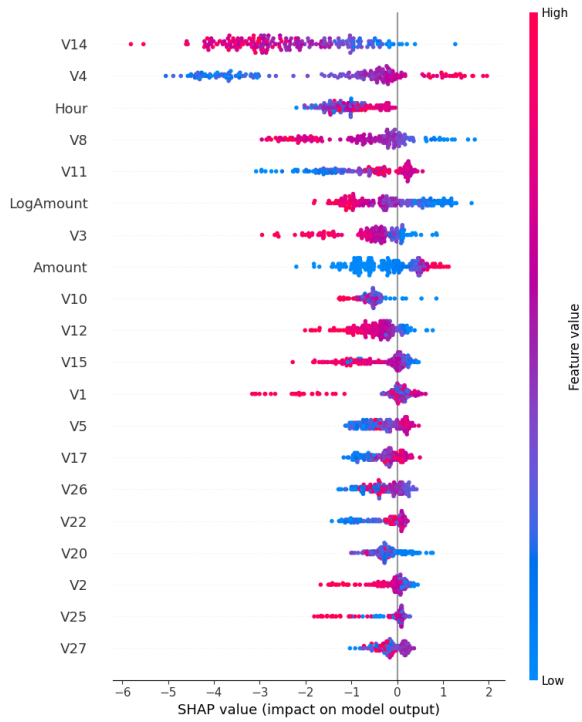


Figure 9: SHAP Summary Plot for Model Interpretability

Table 15: Cost Sensitivity Analysis

FP Cost	FN Cost	Optimal Threshold	Minimum Validation Cost
1	5	0.66	74.0
2	5	0.66	83.0
5	5	0.94	95.0
1	10	0.66	139.0
2	10	0.66	148.0
5	10	0.94	170.0
1	20	0.21	250.0
2	20	0.66	278.0
5	20	0.66	305.0
1	50	0.09	567.0
2	50	0.21	610.0
5	50	0.66	695.0

Table 16: System Performance Metrics

Metric	Value
Benchmark Samples	1000
Total Prediction Time (s)	0.001988
Average Latency (ms)	0.001988
Throughput (transactions/s)	502,942.21

The average latency was 0.001988 ms per transaction, with a throughput of approximately 502,942 transactions per second. These figures represent model inference time only. Actual end-to-end latency in real banking environments also depends on infrastructure, concurrency, database access, communication overhead, and integration with core banking systems. These results therefore indicate the potential for real-time deployment, rather than guaranteeing end-to-end real-time performance in production.

Figure 10 shows the calibration curve of the XGBoost model.

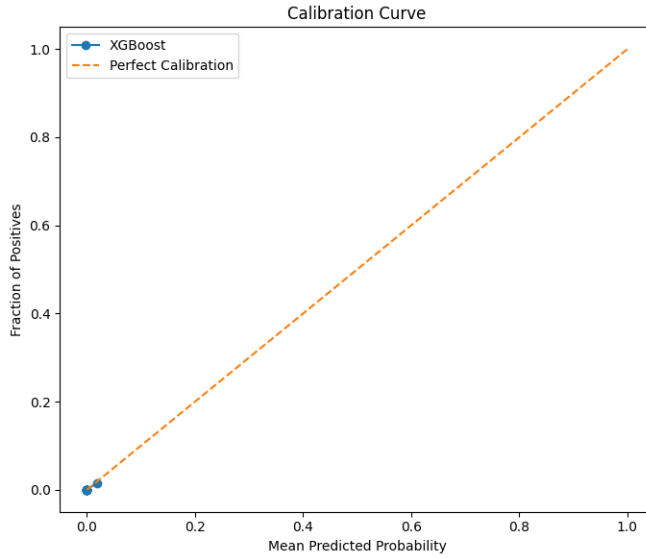


Figure 10: Calibration Curve

5.16 Decision Distribution

Figure 11 shows the distribution of operational decisions on the test set, and Figure 12 shows the MCC as a function of the decision threshold.

6. Discussion

6.1 Interpretation of Results

The experimental results demonstrate that the proposed fraud detection framework achieves strong performance across all evaluation metrics. The ROC-AUC

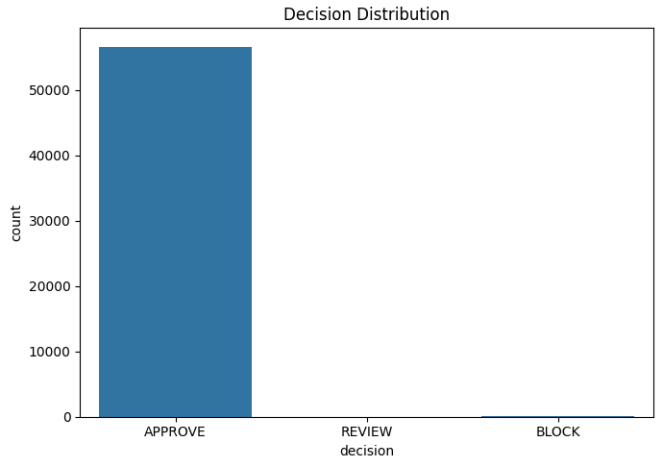


Figure 11: Distribution of Operational Decisions (APPROVE, REVIEW, and BLOCK)

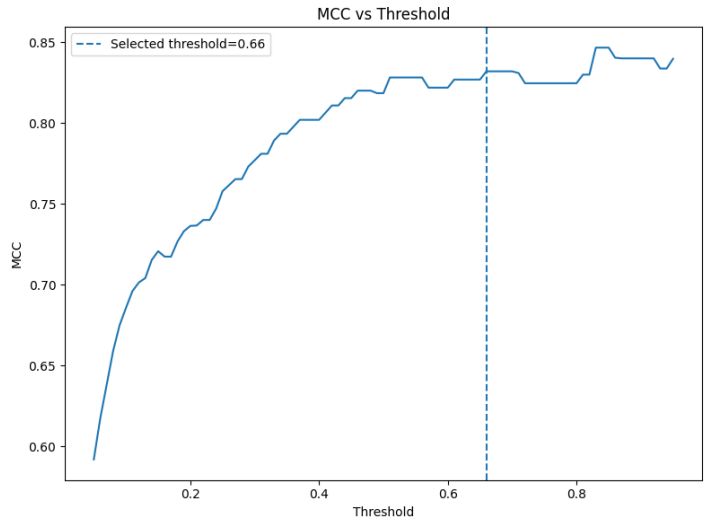


Figure 12: MCC versus Threshold on the Test Set

of 0.9698 indicates excellent discriminative ability, while the PR-AUC of 0.8136 is particularly noteworthy given the severe class imbalance (only 0.167% fraud).

The MCC of 0.8319 confirms that the model maintains balanced performance across both classes, avoiding the common pitfall of high accuracy with poor minority class detection. The Precision of 0.9012 for the fraud class indicates that when the model flags a transaction as fraudulent, it is correct approximately 90% of the time. The Recall of 0.7684 shows that the model successfully detects over 76% of actual fraudulent transactions.

6.2 Comparison with Previous Studies

Compared to previous studies, the proposed framework demonstrates several advantages:

- **Comprehensive System Design:** Unlike studies that focus solely on algorithm comparison, this work presents a complete operational system.
- **Threshold Optimization:** The threshold of 0.66 was selected as the best threshold according to the proposed composite MCC–cost objective on the validation set. It achieves a higher MCC (0.8319 vs. 0.8184) and higher Precision (0.9012 vs. 0.8605) compared to the default threshold of 0.50, at the cost of a slightly lower Recall (0.7684 vs. 0.7789).
- **Multiple Evaluation Metrics:** The inclusion of PR-AUC, MCC, F1-Score, and cost analysis provides a more complete assessment than ROC-AUC alone.
- **Model Interpretability:** SHAP analysis provides transparent explanations for model decisions, which is crucial for regulatory compliance and user trust.
- **System Stability (within the present dataset):** The ablation study and distributional drift analysis between training and test partitions within the present dataset demonstrate that the system architecture is robust. Long-term temporal stability and adaptation to evolving fraud patterns were not evaluated and remain subjects for future research.

6.3 Practical Implications

The proposed framework is designed for real-world deployment in financial institutions. Key practical implications include:

- **Operational Decisions:** The system generates actionable BLOCK, REVIEW, or APPROVE decisions that can be directly integrated with banking systems.
- **Cost-Effective:** The cost-sensitive threshold optimization minimizes the financial impact of classification errors.
- **Real-Time Processing Potential:** With a measured batch-inference latency of approximately 1.99 microseconds per transaction on the benchmark hardware, the model demonstrates computational feasibility for high-throughput scenarios. However, actual end-to-end operational latency in a real banking environment also depends on infrastructure, concurrency, database access, communication overhead, and integration with core banking systems. Therefore, these results indicate the potential for real-time deployment, rather than guaranteeing end-to-end real-time performance in production.
- **Interpretable:** SHAP explanations help fraud analysts understand why specific transactions were flagged.
- **Distributionally Stable in the Present Dataset:** The PSI-based analysis indicates distributional similarity between the training and test partitions in the present dataset. Long-term temporal stability and adaptation to evolving fraud patterns were not evaluated and remain subjects for future research.

6.4 Impact of SMOTE on Model Performance

The ablation study reveals an interesting observation: the model trained without SMOTE achieved slightly higher MCC (0.8444) and F1-score (0.8427) compared to the Full Model (MCC: 0.8336, F1: 0.8242). This result suggests that SMOTE, while widely adopted for handling class imbalance, may introduce slight distributional distortions in some scenarios. However, SMOTE remains an optional imbalance-handling component of the proposed framework, particularly for datasets in which the minority class is extremely scarce. Its effect on generalization may vary across datasets, cost structures, and evaluation criteria. Therefore, SMOTE should be regarded as an important building block of the imbalance-handling pipeline rather than a guaranteed performance enhancer in every setting.

6.5 Limitations

- **Dataset Age:** The dataset originates from 2013, which may not fully represent current fraud patterns.

- **Feature Anonymization:** The PCA-transformed features (V1–V28) limit interpretability of original transaction characteristics.
- **Cost Assumptions:** Cost analysis is based on general assumptions that may vary across different banking environments.
- **Geographic Scope:** The dataset represents a specific European region, which may limit generalizability.
- **No Long-Term Concept-Drift Evaluation:** The present study evaluates distributional drift between training and test partitions using PSI, but does not perform chronological or streaming evaluation. Long-term temporal stability and adaptation to evolving fraud patterns were not evaluated and remain subjects for future research.
- **Heuristic REVIEW Threshold:** The REVIEW threshold is heuristically defined as 70% of the operational threshold, rather than separately optimized. Its operational effectiveness requires further validation.

7. Conclusion

This study presented a system-oriented fraud detection framework integrating preprocessing, feature engineering, optional SMOTE, XGBoost, composite MCC–cost threshold optimization, SHAP explainability, and operational decisions to BLOCK, REVIEW, or APPROVE. The threshold of 0.66 was selected on validation data. Independent test performance reached ROC-AUC 0.9698, PR-AUC 0.8136, and MCC 0.8319. Bootstrap intervals, statistical testing, ablation, cost sensitivity, and PSI analysis provided complementary assessments of performance and limitations.

The measured average batch-inference latency of approximately 1.99 microseconds per transaction and throughput of 502,942 transactions per second indicate computational feasibility on the benchmark hardware. Production latency remains dependent on infrastructure, concurrency, and integration. SHAP supports the interpretation of model predictions, while cost analysis examines different operational requirements.

SMOTE is optional: the ablation study showed dataset-dependent effects on generalization. Low PSI values indicate similarity between the sampled training and test distributions, without establishing long-term temporal stability or adaptation to evolving fraud. Chronological evaluation, concept-drift adaptation, and operational validation remain necessary extensions.

8. Future Work

Despite the promising performance of the proposed framework, several avenues remain for future research:

- **Larger and More Diverse Datasets:** Future studies can evaluate the framework on larger and more diverse financial transaction datasets to further assess its generalization capability across different fraud patterns and operational environments.
- **Cost-Sensitive Learning:** Incorporating cost-sensitive learning strategies could improve the practical effectiveness of the system.
- **Real-Time Streaming Detection:** Extending the framework toward real-time or streaming fraud detection scenarios.
- **Advanced Machine Learning Architectures:** Exploring hybrid or deep learning models to capture complex nonlinear fraud patterns.
- **Enhanced Interpretability:** Incorporating more advanced explainability techniques.
- **Operational Evaluation:** Evaluating the framework within simulated or real operational environments.
- **Federated Learning:** Exploring federated learning approaches for cross-institutional fraud detection.
- **Long-Term Temporal and Concept-Drift Evaluation:** Performing chronological or streaming evaluation to assess long-term stability and adaptation to evolving fraud patterns.

References

Baesens, B., Höppner, S., & Verdonck, T. (2021). Data engineering for fraud detection. *Decision Support Systems*, **150**, 113492.

Breiman, L. (2001). Random Forests. *Machine Learning*, **45**(1), 5–32.

Carcillo, F., Le Borgne, Y.-A., Caelen, O., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, **557**, 159–173.

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

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD*, 785–794.

Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating Probability with Undersampling for Unbalanced Classification. *IEEE Computational Intelligence Magazine*, **10**(4), 20–27.

Ferri, C., Hernández-Orallo, J., & Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, **30**(1), 27–38.

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. *Information Sciences*, **479**, 448–455.

Hasan, M., Rahman, M. S., Janicke, H., & Sarker, I. H. (2024). Detecting Anomalies in Blockchain Transactions using Machine Learning Classifiers and Explainability Analysis. *Blockchain: Research and Applications*, **5**(3), 100207.

Liu, M., Peng, Y., Zhang, Y., Zhang, Y., & Gong, H. (2024). A Hybrid CNN-SVM-LightGBM Model for Enhanced Credit Card Fraud Detection on Imbalanced Datasets. *Proceedings of EITCE '24*, 992–995.

Moder, P., Hoberg, K., & Papier, F. (2026). Decision Threshold Setting in Binary Classification Problems—A Behavioral Lens. *Journal of Operations Management*, **72**(4), 536.

Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, **2**(1), 37–63.

Ullah, F., Babar, M. A., & Aleti, A. (2022). Design and evaluation of adaptive system for big data cyber security analytics. *Expert Systems with Applications*, **207**, 117948.