Vol. 8, No. 4, August 2026

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Interpretable Feature-Scenario Analysis for Ethereum Transaction Anomaly Detection Using Random Forest and XGBoost

Indana Zulfa[1], Kusnawi[2]

[1,2]Department of Informatics, Faculty of Computer Science, Universitas AMIKOM Yogyakarta, Indonesia

Received:
February 2, 2026
Revised:
June 18, 2026
Accepted:
July 22, 2026
Published:
August 22, 2026

Corresponding Author:
Author Name*:
Indana Zulfa
Email*:
indfzla@students.amikom.ac.id

DOI:
10.63158/journalisi.v8i4.1727

Abstract: The substantial and imbalanced volume of Ethereum transactions presents significant challenges for anomaly detection, especially when labels serve as proxies for execution errors rather than confirmed fraud. An interpretable feature-scenario framework was developed utilizing Logistic Regression, Random Forest, and XGBoost on 4,604,555 unique transactions. The isError attribute was employed as a proxy anomaly label. Data splitting occurred prior to address encoding; encoders were trained exclusively on the training set, unseen wallets were assigned a reserved code, and Random Under Sampling (RUS) was applied solely to training data. Evaluation incorporated both an imbalanced random test set and future-block validation. Among 920,911 random-test transactions (3.59% anomalies), Random Forest, excluding the Hour feature and without resampling, achieved optimal operational performance: 0.8204 precision, 0.6716 recall, 0.7386 F1-score, 0.7820 PR-AUC, 0.7338 MCC, and a 0.0055 false-positive rate. Application of RUS increased recall to 0.9035 but reduced precision to 0.3088, resulting in 69.12% of 96,703 alerts being false positives. Future-block validation further reduced PR-AUC to 0.0177 and MCC to 0.0678, indicating a substantial distribution shift. SHAP identified destination-wallet encoding and BlockHeight as the most influential model features, while LIME provided local, non-causal explanations. The primary contribution is an interpretable feature-scenario and validation framework; however, verified malicious labels and dynamic graph representations are still required for operational deployment.

Keywords: Anomaly Detection; Ethereum Transaction Analysis; LIME; Random Forest; SHAP

1. INTRODUCTION

Ethereum processes large numbers of pseudonymous transactions through accounts, externally owned addresses, and programmable smart contracts. Transaction execution failures can arise from insufficient gas, reverted contract logic, invalid calls, or other execution conditions; they do not necessarily indicate illegal or malicious activity. Nevertheless, error-indicating transactions can provide useful signals for screening abnormal behavior and operational instability in Ethereum environments [1], [2].

Existing blockchain anomaly-detection research has applied supervised machine learning, ensemble methods, graph learning, and explainable artificial intelligence (XAI) to illicit-account identification, suspicious-account screening, phishing-node detection, and transaction-level anomalies [3], [4]. These tasks differ in their labels and units of analysis. Illicit-account or phishing labels refer to externally identified malicious entities, whereas the isError attribute used in this study describes transaction execution status. Consequently, the present task is framed as classification of anomaly-indicating or error-related transactions rather than verified fraud detection.

Table 1 summarizes representative Ethereum anomaly and illicit-activity studies. Prior research has demonstrated the value of account-level statistics, graph features, ensemble learning, and XAI, but most studies emphasize algorithm comparison under random holdout evaluation. Fewer studies jointly examine feature-group contribution, training-distribution choices, alert burden under the original class imbalance, and generalization to transactions from later blocks.

Table 1. Comparison of representative Ethereum anomaly-detection studies

Study	Label/task	Features	Model	Validation	limitation/contribution
Farrugia et al. [5]	Illicit-account labels	Account and graph statistics	Supervised ML	Reported holdout evaluation	Account-level task; no feature-scenario/XAI analysis
Aziz et al. [6]	Fraud-labelled accounts	Transaction/account statistics	LightGBM	Random holdout	No temporal validation or model explanations

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Study	Label/task	Features	Model	Validation	limitation/contribution
Hasan et al. [7]	Blockchain anomaly labels	Tabular transaction features	Multiple ML and XAI	Holdout evaluation	Limited analysis of training distribution and alert burden
Ndiaye et al. [8]	Smart-contract behavior anomalies	Contract behavior features	Behavioral anomaly framework	Contract-centric validation	Not a transaction execution-error task
Chen et al. [9]	Phishing-node labels	Graph/network features	Hybrid GNN	Node-classification evaluation	Phishing-specific; limited local explanations
Chithanuru and Ramaiah [10]	Anomalous Ethereum accounts	Engineered account features	Ensemble stacking and XAI	Random holdout	No future-block validation
Gu and Dib [11]	Fraud-labelled Ethereum data	Engineered transaction features	Ensemble learning	Holdout comparison	Focus on classifier performance
Proposed study	isError proxy label	Wallet, value, block, and Hour scenarios	LR baseline, RF, XGBoost and XAI	Original imbalanced random test, no-resampling baseline, and future-block test	Explicit feature scenarios, alert burden, and temporal generalization audit

Class imbalance further complicates evaluation because a high accuracy can coexist with weak minority-class detection. Resampling may increase sensitivity but can also increase false alerts, making precision, PR-AUC, MCC, false-positive rate (FPR), and absolute alert burden essential alongside recall [12]. In addition, address encoding can encourage

memorization of identifiers, while random splitting can mix historical periods and overestimate performance on future transactions.

This study therefore proposes an interpretable feature-scenario framework with three objectives: (1) quantify the contribution of wallet interaction, transaction value, BlockHeight, and Hour features; (2) compare Random Under Sampling with training on the original imbalanced distribution while evaluating on an untouched imbalanced test set; and (3) assess future-block generalization and explain model behavior using SHAP and LIME. The novelty lies in combining feature-scenario analysis, operational alert-burden evaluation, training-strategy comparison, future-block validation, and global/local XAI within one Ethereum transaction analysis pipeline, rather than merely comparing Random Forest and XGBoost.

2. METHODS

Figure 1 presents the complete leakage-free experimental workflow used in this study. The process begins with the collection of 6,794,521 raw Ethereum transaction records, followed by data cleaning and TxHash-based deduplication, which produced 4,604,555 unique transactions. The cleaned data were then partitioned using either a stratified random split or a block-ordered split before any address transformation was performed. This ordering ensured that information from the testing partitions did not influence preprocessing decisions.

After partitioning, address encoders and frequency mappings were fitted exclusively on the training data. Wallet addresses that were not observed during training were represented using the reserved value -1, while frequency encoding was examined as an alternative representation. Two training strategies, no resampling and Random Under Sampling, were evaluated using Logistic Regression as a linear baseline and Random Forest and XGBoost as the main classifiers across four feature scenarios. Model performance was evaluated on the untouched original test distribution and on later blockchain blocks. Finally, SHAP and LIME were employed to provide global and local explanations of the trained models.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

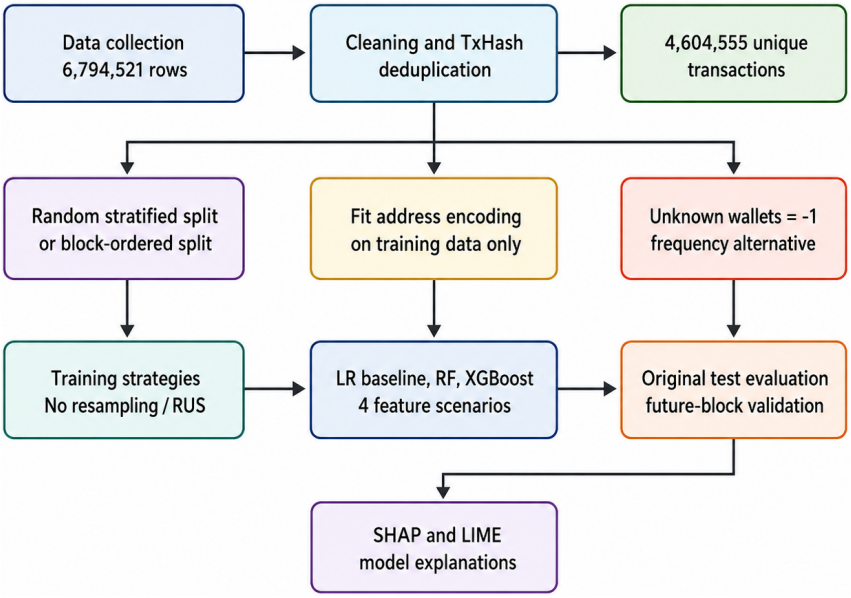

Figure 1. Leakage-free feature-scenario and validation workflow

The sequence in Figure 1 is central to the experimental design because preprocessing, address representation, and resampling are learned only from training observations. The two validation branches distinguish performance under an original imbalanced random holdout from generalization to later blockchain blocks, while the final explainability stage is applied only after model selection and evaluation.

2.1. Dataset and proxy label

The publicly available dataset entitled "Ethereum Transactions for Fraud Detection" contains TxHash, BlockHeight, TimeStamp, From, To, Value, and isError attributes. The raw file contained 6,794,521 rows. Rows missing essential numerical or target fields were removed; no such rows were found. Missing sender and receiver addresses were retained using the explicit tokens __UNKNOWN_FROM__ and __UNKNOWN_TO__. Duplicate transactions were then removed using TxHash, leaving 4,604,555 unique records. The final distribution comprised 4,439,278 successfully executed transactions (96.41%) and 165,277 error-indicating transactions (3.59%). The positive class is therefore a proxy for transaction execution failure or anomaly-indicating behavior, not an externally verified fraud label. Transaction- and account-level Ethereum attributes of this type have also been used in prior machine-learning anomaly studies [5], [7].

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

To illustrate the structure and content of the transaction records, Table 2 presents six representative samples selected from the supplied dataset extract. The examples include three successfully executed transactions and all three transactions marked by the isError attribute. For readability, transaction hashes and wallet addresses are abbreviated in the manuscript, while the complete identifiers were retained during preprocessing and model development.

Table 2. Representative samples of Ethereum transaction records

TxHash	BlockHeight	TimeStamp	From	To	Value	isError
0x90927658…5c5b2f	5,629,924	1,526,570,799	0xf5bec430…0884f3	0x3f37b983…d18a0b	0.34642742	0
0x00979508…0cda03	6,180,479	1,534,755,063	0x52f37dfc…a7299c	0xc867c3e1…f2071c	17.99991600	0
0x58b88a41…ee6f0f	6,926,794	1,545,397,592	0x3fbed3c3…805576	0x5122df80…513934	1.59733885	0
0xd067e54b…40f4a1	3,909,955	1,498,080,861	0xac143ac0…678e6b	0x55d34b68…a00a4c	1.00000000	1
0x11ada702…59354a	3,933,540	1,498,493,139	0x3fef41a4…e92844	0x5678bcea…f0a2b1	4.20000000	1
0xa512ecf8…de5c09	4,224,452	1,504,209,569	0xe6eebf59…39fcdd	0x741fc999…8ef24e	3.00000000	1

As shown in Table 2, each record contains a unique transaction hash, its blockchain position, Unix timestamp, sender and destination wallet addresses, transferred value, and execution-status label. An isError value of 0 denotes a successfully executed transaction, whereas a value of 1 indicates that the transaction encountered an execution error. The positive class is used only as an anomaly-indicating proxy and is not evidence of fraudulent or malicious activity. The selected dataset attributes, their data types, and the treatment applied during preprocessing are summarized in Table 3. This table also distinguishes fields used only for deduplication from features retained for modeling and the proxy target used during evaluation.

Table 3. Dataset attributes and preprocessing treatment

Attribute	Type	Use/treatment
TxHash	String	Deduplication key; excluded from modeling
BlockHeight	Integer	Structural and block-order feature
TimeStamp	Integer	Converted to UTC datetime; Hour derived
From	String	Missing values mapped to __UNKNOWN_FROM__
To	String	Missing values mapped to __UNKNOWN_TO__
Value	Double	Transaction-value feature; extreme values retained
isError	Integer	Proxy target: 0 = successful, 1 = execution error/anomaly-indicating

Table 3 shows that TxHash is used only as a deduplication key, while BlockHeight, Hour derived from TimeStamp, wallet endpoints, and transaction Value form the candidate predictor groups. Missing wallet values are retained through explicit unknown tokens, and isError is maintained as the binary proxy target rather than being interpreted as a verified fraud label.

2.2. Data partitioning and wallet representation

For the primary evaluation, the cleaned dataset was divided using an 80:20 stratified split with random_state = 42. The training set contained 3,683,644 transactions, including 132,222 positive cases; the untouched test set contained 920,911 transactions, including 33,055 positive cases (3.589%). The address encoder was fitted only on the training From and to values using OrdinalEncoder(handle_unknown="use_encoded_value", unknown_value=-1). Test or deployment addresses not observed during training were therefore assigned −1. In the random test set, 11.42% of sender addresses and 12.48% of destination addresses were unseen. Training-only frequency mappings were also evaluated as a robustness alternative; unseen addresses received a frequency of 0.

A second validation design ordered records by BlockHeight, timestamp, and TxHash. The first 80% of unique blocks were used for training and later blocks were reserved for testing. The training partition contained blocks 46,230–7,131,601, whereas the future-block

test contained blocks 7,131,603–8,754,331. This design assesses distribution shift and avoids mixing later block periods into model training. This block-ordered design follows the temporal-validation motivation of Ethereum studies that evaluate behavior across distinct blockchain periods [13], [14]. More generally, blocked cross-validation is recommended when temporal dependence can make random holdout estimates overly optimistic [15].

2.3. Class-imbalance strategies

RandomUnderSampler from imbalanced-learn was configured with sampling_strategy="auto", random_state=42, and replacement=False. It was applied only to the training partition and reduced the random-split training set to 132,222 normal and 132,222 anomaly-indicating transactions. A no-resampling baseline retained all 3,683,644 training records. Comparing these strategies quantifies the trade-off between minority-class sensitivity and operational false-alert burden. Random Under Sampling was selected to reduce majority-class dominance without synthesizing minority observations, although it may discard informative normal examples [12]. Comparative studies of rebalancing methods likewise show that the effect of sampling depends on the class distribution and the learning algorithm [16].

2.4. Feature scenarios

To isolate the contribution of each feature group, four experimental configurations were constructed, as summarized in Table 4. The scenarios extend prior transaction-feature and explainability analyses [7], [10] by explicitly removing address identifiers or the short-term Hour feature while preserving an otherwise comparable evaluation pipeline.

Table 4. Experimental feature scenarios

Scenario	Features	Purpose
Baseline	From_encoded, To_encoded, Value	Wallet endpoints and transaction value
Full Features	From_encoded, To_encoded, Value, BlockHeight, Hour	All available features
Without Address	Value, BlockHeight, Hour	Contribution of wallet identifiers

Scenario	Features	Purpose
Without Hour	From_encoded, To_encoded, Value, BlockHeight	Removes only the explicit short-term Hour feature

The former "Without Temporal" label was renamed "Without Hour" because BlockHeight remains in the scenario and may encode temporal ordering. BlockHeight is treated as structural block-position information, but its interpretation is explicitly qualified because block numbers increase over time and can capture historical dataset structure.

2.5. Models and implementation

2.5.1. Logistic Regression baseline

Logistic Regression was included only as a linear baseline. It used the lbfgs solver, max_iter=1000, class_weight=None, and random_state=42. The posterior probability is expressed as shown in Equation 1.

$$P(y = 1 \mid x) = 1 / [1 + \exp\{-(\beta_0 + \beta^T x)\}] \quad (1)$$

Equation (1) transforms the linear combination of the input features into a probability between zero and one. The resulting probability was converted into a binary class prediction using the default classification threshold, allowing Logistic Regression to serve as a transparent linear baseline.

2.5.2. Random Forest

Random Forest combines multiple decision trees and predicts the class receiving the majority vote [17]. It was configured with n_estimators=50, random_state=42, and n_jobs=−1, shown in Equation 2.

$$\hat{y} = \text{mode}\{T_i(x)\}_{i=1}^{n} \quad (2)$$

As expressed in Equation (2), the Random Forest prediction is determined from the majority class selected by the individual decision trees. Combining multiple trees reduces

the instability associated with relying on a single decision tree and provides a nonlinear comparison with the Logistic Regression baseline.

2.5.3. XGBoost

XGBoost constructs additive trees by minimizing a regularized objective [18]. The model used n_estimators=100, max_depth=6, learning_rate=0.1, subsample=0.8, colsample_bytree=0.8, tree_method="hist", eval_metric="logloss", random_state=42, and n_jobs=−1, shown in Equation 3.

$$L = \Sigma_i \, l\,(y_i, \hat{y}_i) + \Sigma_k \, \Omega(f_k) \quad (3)$$

Equation (3) shows that XGBoost minimizes both the prediction loss and a regularization term. The regularization component constrains model complexity, while the additive tree construction iteratively corrects errors made by preceding trees.

To support reproducibility, the execution environment, package versions, model settings, and global random seed used in the experiments are presented in Table 5. Reporting these details enables the leakage-free pipeline and the model comparisons to be replicated under a consistent software configuration.

Table 5. Software environment and reproducibility settings

Component	Version/configuration
Execution environment	Google Colab; Linux 6.6.122+ x86_64
Python	3.12.13
pandas / NumPy	2.2.2 / 2.0.2
scikit-learn	1.6.1
imbalanced-learn	0.14.2
XGBoost	3.3.0
SHAP	0.52.0
LIME	0.2.0.1
Global random seed	42

Table 5 confirms that all experiments were executed in Google Colab using Python 3.12.13 and a fixed random seed of 42. The reported library versions correspond to the

environment used to regenerate the final metrics, confusion matrices, SHAP values, and LIME explanation included in this manuscript.

2.6. Evaluation metrics and model selection

All reported random-split metrics were computed from the same untouched test set. The classification measures used in this study are formally defined in Equations (4)–(11), where TP, FP, TN, and FN denote true positives, false positives, true negatives, and false negatives, respectively [19]. Each equation captures a different aspect of model behavior under severe class imbalance.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (4)$$

$$\text{Precision} = TP / (TP + FP) \quad (5)$$

$$\text{Recall} = TP / (TP + FN) \quad (6)$$

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (7)$$

$$\text{Specificity} = TN / (TN + FP) \quad (8)$$

$$FPR = FP / (FP + TN) = 1 - \text{Specificity} \quad (9)$$

$$\text{Balanced Accuracy} = \tfrac{1}{2} [TP/(TP+FN) + TN/(TN+FP)] \quad (10)$$

$$MCC = (TP \times TN - FP \times FN) / \sqrt{[(TP+FP)\,(TP+FN)\,(TN+FP)\,(TN+FN)]} \quad (11)$$

Equations (4)–(11) were interpreted jointly rather than in isolation. Precision and FPR quantify the reliability and operational burden of anomaly alerts, Recall measures sensitivity to the minority class, and F1-score balances Precision and Recall. Balanced Accuracy and MCC provide distribution-aware summaries that remain informative when the normal class dominates the dataset. MCC was emphasized because it summarizes all four confusion-matrix outcomes and is generally more reliable than accuracy or F1-score alone on imbalanced binary tasks [20].

ROC-AUC and PR-AUC were computed from predicted probabilities. Because PR-AUC for a non-informative classifier equals the positive prevalence, PR-AUC lift was calculated as PR-AUC divided by anomaly prevalence. Absolute alert count (TP+FP), false-alert share FP/(TP+FP), and alerts per 100,000 transactions were also reported. The operationally balanced model was selected primarily by MCC, supported by F1-score, PR-AUC, precision, and FPR. A separate sensitivity-oriented configuration was retained when recall was the

principal objective. PR-AUC was emphasized because precision–recall analysis is more informative than ROC analysis when the positive class is rare [21].

2.7. Explainability analysis

SHAP was used to summarize global feature contributions for the sensitivity-oriented Random Forest Without Hour model [22]. LIME was applied to a true-positive test instance to provide a local approximation of the model decision [23]. Both methods were interpreted as explanations of the trained model's behavior, not as evidence that a feature causally produces an execution error or malicious transaction. This post-hoc interpretation follows broader XAI guidance that explanations should be assessed in relation to their intended audience, scope, and limitations [24].

3. RESULTS AND DISCUSSION

3.1. Data audit and validation partitions

The number of observations, class distributions, anomaly prevalence values, and block ranges produced by cleaning and both validation designs are summarized in Table 6. These counts establish the population on which all subsequent performance and alert-burden results are based.

Table 6. Data-cleaning and validation summary

Stage/partition	Rows	Normal	Anomaly	Prevalence	Block range
Raw dataset	6,794,521	—	—	—	—
After TxHash deduplication	4,604,555	4,439,278	165,277	3.589%	46,230–8,754,331
Random training set	3,683,644	3,551,422	132,222	3.589%	46,230–8,754,151
Random test set	920,911	887,856	33,055	3.589%	46,240–8,754,331
Future-block training	3,830,958	3,670,110	160,848	4.199%	46,230–7,131,601
Future-block test	773,597	769,168	4,429	0.573%	7,131,603–8,754,331

The future-block test had substantially lower anomaly prevalence than the random test and contained 33.44% unseen sender wallets and 38.59% unseen destination wallets. This shift provides a stricter assessment of future-period generalization.

3.2. Random-split performance with Random Under Sampling

The first model-comparison experiment applied Random Under Sampling only to the training partition and retained the original imbalanced test set. Table 7 reports the resulting performance for Logistic Regression, Random Forest, and XGBoost across all four feature scenarios.

Table 7. Random-stratified test results after Random Under Sampling of training data

Model	Scenario	Acc.	Prec.	Recall	F1	PR-AUC	MCC	FPR
Logistic Regression	Baseline	0.487	0.043	0.628	0.081	0.042	0.041	0.518
Logistic Regression	Full Features	0.693	0.070	0.614	0.125	0.071	0.124	0.304
Logistic Regression	Without Address	0.662	0.066	0.643	0.120	0.063	0.119	0.337
Logistic Regression	Without Hour	0.604	0.067	0.774	0.123	0.069	0.140	0.403
Random Forest	Baseline	0.868	0.194	0.849	0.316	0.494	0.367	0.131
Random Forest	Full Features	0.925	0.309	0.889	0.459	0.674	0.498	0.074
Random Forest	Without Address	0.843	0.167	0.841	0.278	0.438	0.331	0.157
Random Forest	Without Hour	0.924	0.309	0.904	0.460	0.674	0.503	0.075
XGBoost	Baseline	0.867	0.179	0.755	0.290	0.410	0.325	0.129
XGBoost	Full Features	0.868	0.191	0.831	0.311	0.520	0.359	0.131
XGBoost	Without Address	0.810	0.135	0.792	0.230	0.410	0.275	0.190
XGBoost	Without Hour	0.862	0.184	0.834	0.302	0.517	0.351	0.137

Among RUS-trained models, Random Forest produced the strongest results. The Without Hour scenario achieved recall of 0.9035, PR-AUC of 0.6735, and MCC of 0.5026. Removing wallet-address features reduced Random Forest MCC to 0.3307, confirming that endpoint identifiers contributed substantially to the learned decision rules. Full Features and

Without Hour performed similarly, indicating that Hour added limited information under the random split.

3.3. No-resampling baseline and training-strategy trade-off

To determine whether the sensitivity gained through Random Under Sampling justified its operational cost, the strongest Random Forest and XGBoost scenarios were retrained without resampling under the same untouched test conditions. The direct training-strategy comparison is presented in Table 8.

Table 8. Comparison of Random Under Sampling and no-resampling training

Model	Scenario	Training	Prec.	Recall	F1	PR-AUC	MCC	FPR
Random Forest	Full Features	No Resampling	0.836	0.642	0.726	0.771	0.724	0.0047
Random Forest	Full Features	Random Under Sampling	0.309	0.889	0.459	0.674	0.498	0.0740
Random Forest	Without Hour	No Resampli ng	0.820	0.672	0.739	0.782	0.734	0.0055
Random Forest	Without Hour	Random Under Sampling	0.309	0.904	0.460	0.674	0.503	0.0753
XGBoost	Full Features	No Resampling	0.834	0.262	0.398	0.548	0.458	0.0019
XGBoost	Full Features	Random Under Sampling	0.191	0.831	0.311	0.520	0.359	0.1311
XGBoost	Without Hour	No Resampling	0.856	0.236	0.370	0.548	0.441	0.0015
XGBoost	Without Hour	Random Under Sampling	0.184	0.834	0.302	0.517	0.351	0.1373

The no-resampling Random Forest Without Hour model achieved the strongest operational balance: precision 0.8204, recall 0.6716, F1-score 0.7386, PR-AUC 0.7820, MCC 0.7338, and FPR 0.0055. The RUS counterpart increased recall to 0.9035 but reduced

precision to 0.3088 and MCC to 0.5026. Thus, RUS is appropriate when missing an anomaly-indicating transaction is considered more costly than generating false alerts, whereas no resampling is preferable when analyst workload and alert precision are central.

3.4. Confusion matrices and operational alert burden

To examine classification errors in absolute numbers, Figure 2 presents the confusion matrix of the sensitivity-oriented Random Forest Without Hour model, whereas Figure 3 presents the corresponding result for XGBoost Full Features. Both models were trained using Random Under Sampling and evaluated on the same untouched imbalanced random test set.

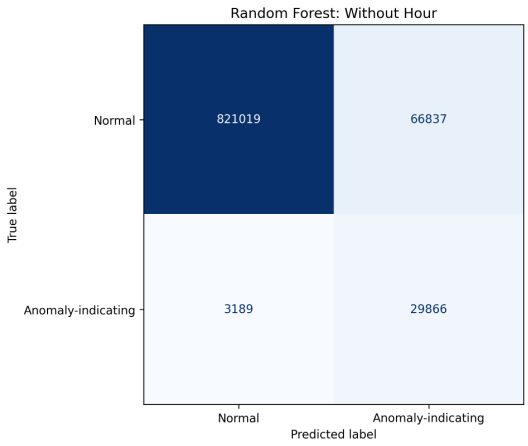

Figure 2. Confusion matrix of Random Forest Without Hour with Random Under Sampling

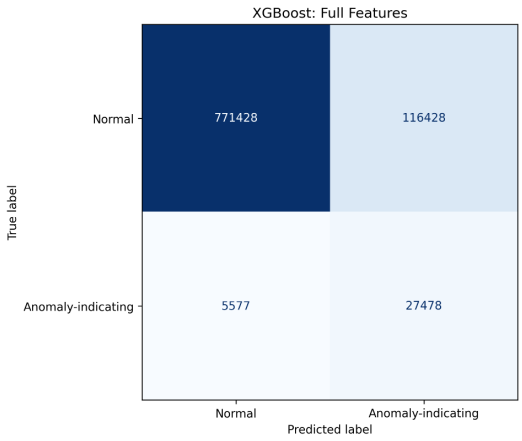

Figure 3. Confusion matrix of XGBoost Full Features with Random Under Sampling

As shown in Figure 2, Random Forest correctly identified 821,019 normal transactions and 29,866 anomaly-indicating transactions, while producing 66,837 false positives and 3,189 false negatives. Figure 3 shows that XGBoost correctly classified 771,428 normal transactions and 27,478 anomaly-indicating transactions but generated 116,428 false positives and 5,577 false negatives. The lower error counts of Random Forest are consistent with its stronger Precision, Recall, F1-score, and MCC, and they motivate the operational alert-burden comparison reported in Table 9.

Because confusion-matrix rates can understate workload in a dataset containing hundreds of thousands of normal transactions, Table 9 converts the selected Random Forest results into absolute alerts, false-alert shares, and alerts per 100,000 transactions. This presentation links statistical performance to the volume of cases that would require analyst review.

Table 9. Operational alert burden for selected Random Forest Without Hour configurations

Training	TN	FP	FN	TP	Alerts	False-alert share	Alerts/100k
No resampling	882,997	4,859	10,855	22,200	27,059	17.96%	2,938
Random Under Sampling	821,019	66,837	3,189	29,866	96,703	69.12%	10,501

Although the RUS model detected 29,866 of 33,055 positive cases, it generated 66,837 false positives. Approximately 69% of its alerts were therefore false, compared with 17.96% for the no-resampling model. The 7.53% FPR of the RUS model may appear numerically moderate but applied to 887,856 normal test transactions it created a substantial analyst burden. This result prevents the RUS configuration from being described as deployment-ready despite its high recall. The anomaly prevalence baseline for PR-AUC was 0.0359. Random Forest Without Hour achieved PR-AUC lifts of 18.76× with RUS and 21.79× without resampling, confirming meaningful ranking performance beyond the prevalence baseline.

3.5. Future-block validation

Random-holdout performance was then challenged using a stricter future-block design in which later block ranges were excluded from model training. Table 10 reports the resulting Precision, Recall, F1-score, PR-AUC, MCC, FPR, and false-alert proportions under this temporally separated evaluation. Performance declined sharply when models trained on earlier blocks were evaluated on later blocks. The best future-block result was Random Forest Without Hour, with PR-AUC 0.0177, MCC 0.0678, and recall 0.0768. Although its PR-AUC remained 3.09 times the future-period prevalence baseline of 0.0057, it was far below the random-split result. This demonstrates substantial temporal and population

shift and shows that random holdout performance cannot be interpreted as evidence of stable future deployment.

Table 10. Future-block validation results

Model	Scenario	Prec.	Recall	F1	PR-AUC	MCC	FPR	False alerts
Random Forest	Full Features	0.038	0.061	0.047	0.0172	0.041	0.0088	96.2%
Random Forest	Without Hour	0.070	0.077	0.073	0.0177	0.068	0.0059	93.0%
XGBoost	Full Features	0.056	0.016	0.024	0.0131	0.027	0.0015	94.4%
XGBoost	Without Hour	0.055	0.030	0.038	0.0141	0.036	0.0029	94.5%

3.6. Frequency-encoding robustness check

To assess whether the ordinal representation of wallet identifiers was driving the results, a training-only frequency-encoding alternative was evaluated with Random Under Sampling. Table 11 summarizes this robustness check for Random Forest and XGBoost using the Full Features and Without Hour scenarios.

Table 11. Frequency-encoding robustness results with Random Under Sampling

Model	Scenario	Prec.	Recall	F1	PR-AUC	MCC
Random Forest	Frequency Full Features	0.301	0.904	0.452	0.669	0.495
Random Forest	Frequency Without Hour	0.303	0.909	0.455	0.645	0.499
XGBoost	Frequency Full Features	0.207	0.866	0.334	0.587	0.387
XGBoost	Frequency Without Hour	0.204	0.869	0.330	0.582	0.384

Frequency encoding handled unseen wallets naturally through a zero-frequency value and reduced reliance on arbitrary ordinal codes. However, its Random Forest MCC remained approximately 0.50 and false-alert shares remained close to 70%. It therefore improved deployment semantics but did not resolve the principal precision–recall trade-off.

3.7. SHAP and LIME explanations

After predictive performance was evaluated, SHAP was applied to the sensitivity-oriented Random Forest Without Hour configuration to identify the global patterns used by the fitted model. The distribution and direction of the resulting feature contributions are presented in Figure 4.

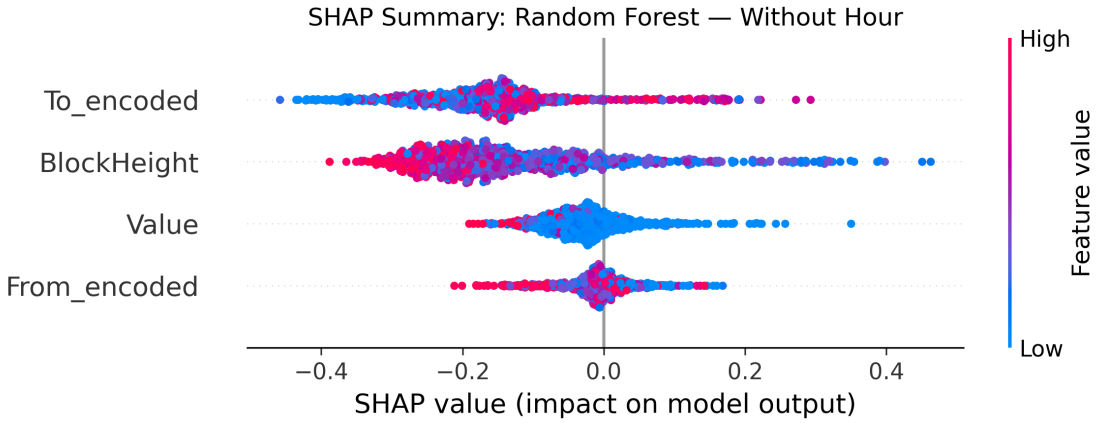

Figure 4. SHAP summary for the Random Forest Without Hour sensitivity-oriented model

To complement the visual distribution in Figure 4 with a concise global ranking, the mean absolute SHAP values of the four retained features are reported in Table 12. The ranking indicates the magnitude with which the trained model used each feature, not a causal effect on transaction execution.

Table 12. Mean absolute SHAP importance

Feature	Mean |SHAP|
To_encoded	0.1759
BlockHeight	0.1656
Value	0.0486
From_encoded	0.0313

To_encoded had the largest mean absolute SHAP value (0.1759), followed by BlockHeight (0.1656), Value (0.0486), and From_encoded (0.0313). The result differs from the previous draft, which incorrectly described BlockHeight as the dominant feature. Because address encodings are identifiers rather than quantities, these rankings indicate how the fitted

model used partitions of the encoded feature space; they do not imply that a numerically larger address is more anomalous. Because global SHAP importance does not explain how the model reaches a specific decision, LIME was subsequently applied to one correctly identified anomaly-indicating transaction. The local positive and negative feature contributions for this instance are visualized in Figure 5.

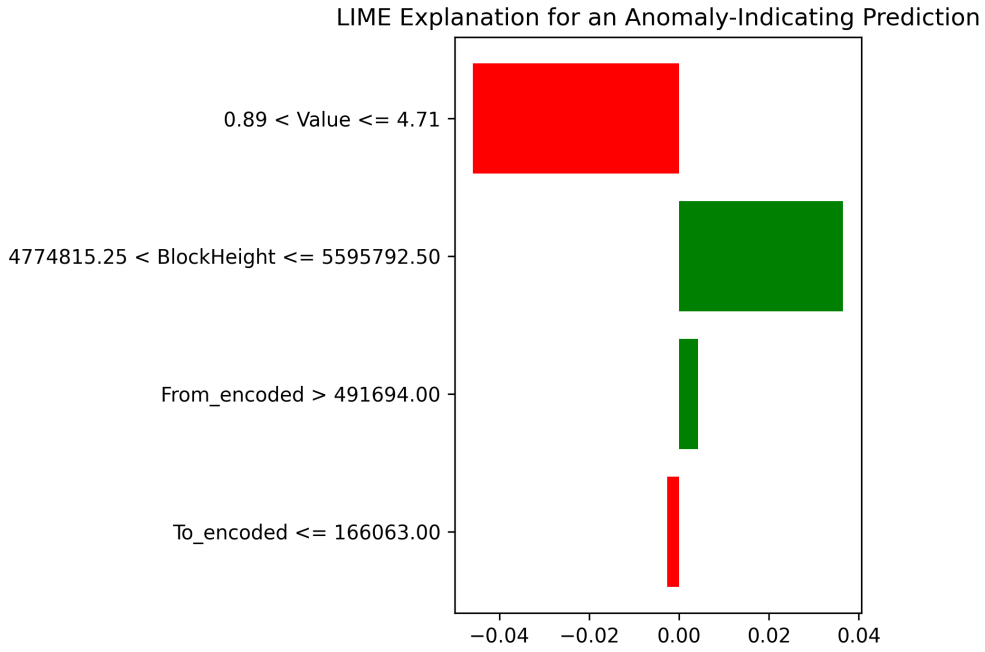

Figure 5. LIME explanation for a correctly detected anomaly-indicating transaction

The numerical conditions and contribution weights corresponding to the bars in Figure 5 are listed in Table 13. Presenting both forms makes the local explanation auditable while preserving the caution that encoded-address thresholds are model-specific rather than universally meaningful wallet boundaries.

Table 13. LIME local contributions for the selected true-positive instance

Local condition	Contribution
0.89 < Value <= 4.71	-0.0459
4774815.25 < BlockHeight <= 5595792.50	+0.0365
From_encoded > 491694.00	+0.0042
To_encoded <= 166063.00	-0.0026

The selected transaction had a true label of 1, a predicted label of 1, and an anomaly probability of 0.94. Its local explanation combined positive and negative contributions from BlockHeight, Value, and encoded wallet endpoints. These rules explain the model around one instance and must not be treated as causal evidence or universally meaningful wallet thresholds.

3.8. Discussion

The revised experiments change the interpretation of model superiority. Random Forest remained stronger than Logistic Regression and XGBoost across the main feature scenarios, but no single training strategy dominated all objectives. Random Under Sampling produced a high-sensitivity profile by detecting approximately 90% of proxy-positive transactions. Training on the original distribution produced a more operationally balanced profile, with substantially higher precision, F1-score, PR-AUC, and MCC and a much lower FPR. Accordingly, model selection should depend on whether the application prioritizes missed anomalies or analyst workload rather than relying on accuracy alone.

Precision of 0.3088 for the RUS model means that fewer than one in three alerts corresponded to an isError-positive transaction. In absolute terms, 66,837 normal transactions were flagged, creating roughly 10,501 alerts per 100,000 transactions. This alert burden would require secondary filtering, calibrated thresholds, analyst triage, or cost-sensitive decision rules before operational use. The no-resampling model reduced the burden to 2,938 alerts per 100,000, but its lower recall left 10,855 positive test transactions undetected. Both profiles therefore involve explicit operational trade-offs.

Feature-scenario results show that removing addresses reduced performance, but this finding must be interpreted cautiously. Ordinal address encoding can enable tree models to memorize recurring wallet identifiers and split on arbitrary numeric partitions. Training-only fitting and unknown code −1 prevent vocabulary leakage, but they do not eliminate memorization. The 11–12% unseen-address rates in random testing and 33–39% rates in future-block testing illustrate why dynamic address representations are needed. Frequency encoding was more semantically appropriate but did not substantially improve discrimination. Graph-degree, transaction-history, neighborhood, and temporal behavioral features are more promising because Ethereum transactions are naturally relational.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Dynamic graph models that evolve with changing node sets provide one possible direction for representing new wallets and temporal network structure [25].

BlockHeight importance may represent genuine changes in execution conditions, network behavior, or contract activity across blockchain periods. It may also capture historical shifts in the proxy label or dataset construction. The severe performance decline in future-block validation supports the latter concern and indicates that BlockHeight can act as a historical shortcut. BlockHeight importance may represent genuine changes in execution conditions, network behavior, or contract activity across blockchain periods. It may also capture historical shifts in the proxy label or dataset construction. The severe performance decline in future-block validation supports the latter concern and indicates that BlockHeight can act as a historical shortcut. Future models should therefore use rolling validation, periodic recalibration, and explicit drift monitoring rather than assuming that the transaction distribution remains stationary. Previous research has demonstrated that distribution-based drift detection combined with event-triggered SHAP can support interpretable diagnosis under non-stationary streaming conditions [26]. In the Ethereum context, this principle could be extended using graph and behavioral features that are recomputed as new blocks and wallet interactions emerge.

SHAP and LIME increased transparency by showing which inputs the trained model used globally and locally, consistent with the role of post-hoc XAI in blockchain analytics [7], [10]. Nevertheless, neither method establishes that a wallet, block height, or value caused an execution error. The explanations are conditional on the proxy target, encoded representation, training period, and model architecture.

The study has five principal limitations. First, isError is a proxy for execution status rather than externally verified malicious activity. Second, random splitting can overestimate future-period performance, as confirmed by block validation. Third, Random Under Sampling discards most normal transactions and changes the training distribution. Fourth, ordinal wallet encoding can encourage address memorization and requires explicit handling of unseen values. Fifth, the block-based experiment revealed strong distribution shift, so the reported random-split metrics should not be generalized as real-world fraud-detection readiness. Stronger future designs should integrate externally

verified labels, dynamic graph representations, threshold calibration, cost-sensitive learning, and longitudinal validation [13], [14].

4. CONCLUSION

This study developed an interpretable feature-scenario framework for classifying Ethereum transactions labelled by isError, which represents execution-error or anomaly-indicating behavior rather than verified fraud. Using a leakage-free pipeline, the study compared Logistic Regression, Random Forest, and XGBoost across four feature scenarios, two training-distribution strategies, an untouched imbalanced random test, and a future-block test. Random Forest Without Hour trained on the original distribution provided the best operational balance on the random test, with precision 0.8204, recall 0.6716, F1-score 0.7386, PR-AUC 0.7820, MCC 0.7338, and FPR 0.0055, whereas Random Under Sampling increased recall to 0.9035 but reduced precision to 0.3088 and generated 66,837 false-positive alerts. The future-block decline to PR-AUC 0.0177 and MCC 0.0678 demonstrates that the findings are valid only within the evaluated settings and do not establish robust performance on future Ethereum activity. SHAP and LIME improved transparency but explained model behavior rather than causality, while the importance of encoded addresses and BlockHeight revealed risks of address memorization and historical shortcuts. Future work should prioritize externally verified malicious-activity labels, graph and behavioral wallet representations, dynamic unseen-address handling, calibrated or cost-sensitive alert thresholds, and rolling temporal or block-based validation before operational deployment.

ACKNOWLEDGMENT

The authors acknowledge the Department of Informatics, Faculty of Computer Science, Universitas AMIKOM Yogyakarta, for supporting the computational experiments conducted in this study.

REFERENCES

[1] N. T. Anthony, M. Shafik, F. Kurugollu, and H. F. Atlam, "Anomaly Detection System for Ethereum Blockchain Using Machine Learning," in *Volume 25: Advances in Manufacturing Technology XXXV*, 2022. doi: 10.3233/ATDE220608.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[2] M. Kezadri Hamiaz and M. Driss, "Ethereum Smart Contracts Under Scrutiny: A Survey of Security Verification Tools, Techniques, and Challenges," *Computers*, vol. 14, no. 6, p. 226, Jun. 2025, doi: 10.3390/computers14060226.

[3] F. Jumani and M. Raza, "Machine Learning for Anomaly Detection in Blockchain: A Critical Analysis, Empirical Validation, and Future Outlook," *Computers*, vol. 14, no. 7, p. 247, Jun. 2025, doi: 10.3390/computers14070247.

[4] R. Shevchuk, V. Martsenyuk, B. Adamyk, V. Benson, and A. Melnyk, "Anomaly Detection in Blockchain: A Systematic Review of Trends, Challenges, and Future Directions," *Applied Sciences*, vol. 15, no. 15, p. 8330, Jul. 2025, doi: 10.3390/app15158330.

[5] S. Farrugia, J. Ellul, and G. Azzopardi, "Detection of illicit accounts over the Ethereum blockchain," *Expert Syst. Appl.*, vol. 150, p. 113318, Jul. 2020, doi: 10.1016/j.eswa.2020.113318.

[6] R. M. Aziz, M. F. Baluch, S. Patel, and A. H. Ganie, "LGBM: a machine learning approach for Ethereum fraud detection," *International Journal of Information Technology*, vol. 14, no. 7, pp. 3321–3331, Dec. 2022, doi: 10.1007/s41870-022-00864-6.

[7] M. Hasan, M. S. Rahman, H. Janicke, and I. H. Sarker, "Detecting anomalies in blockchain transactions using machine learning classifiers and explainability analysis," *Blockchain: Research and Applications*, vol. 5, no. 3, p. 100207, Sep. 2024, doi: 10.1016/j.bcra.2024.100207.

[8] M. Ndiaye, T. A. Diallo, and K. Konate, "ADEFGuard: Anomaly detection framework based on Ethereum smart contracts behaviours," *Blockchain: Research and Applications*, vol. 4, no. 3, p. 100148, Sep. 2023, doi: 10.1016/j.bcra.2023.100148.

[9] Z. Chen, S.-Z. Liu, J. Huang, Y.-H. Xiu, H. Zhang, and H.-X. Long, "Ethereum Phishing Scam Detection Based on Data Augmentation Method and Hybrid Graph Neural Network Model," *Sensors*, vol. 24, no. 12, p. 4022, Jun. 2024, doi: 10.3390/s24124022.

[10] V. Chithanuru and M. Ramaiah, "Proactive detection of anomalous behavior in Ethereum accounts using XAI-enabled ensemble stacking with Bayesian optimization," *PeerJ Comput. Sci.*, vol. 11, p. e2630, Mar. 2025, doi: 10.7717/peerj-cs.2630.

[11] Z. Gu and O. Dib, "Enhancing fraud detection in the Ethereum blockchain using ensemble learning," *PeerJ Comput. Sci.*, vol. 11, p. e2716, Feb. 2025, doi: 10.7717/peerj-cs.2716.

[12] Haibo He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[13] B. Pan, N. Stakhanova, and Z. Zhu, "EtherShield: Time-interval Analysis for Detection of Malicious Behavior on Ethereum," *ACM Trans. Internet Technol.*, vol. 24, no. 1, pp. 1–30, Feb. 2024, doi: 10.1145/3633514.

[14] B. Han, Y. Wei, Q. Wang, F. M. De Collibus, and C. J. Tessone, "MT2AD: multi-layer temporal transaction anomaly detection in ethereum networks with GNN," *Complex & Intelligent Systems*, vol. 10, no. 1, pp. 613–626, Feb. 2024, doi: 10.1007/s40747-023-01126-z.

[15] D. R. Roberts *et al.*, "Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure," *Ecography*, vol. 40, no. 8, pp. 913–929, Aug. 2017, doi: 10.1111/ecog.02881.

[16] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, Jun. 2004, doi: 10.1145/1007730.1007735.

[17] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[18] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[19] S. Sathyanarayanan, "Confusion Matrix-Based Performance Evaluation Metrics," *African Journal of Biomedical Research*, pp. 4023–4031, Nov. 2024, doi: 10.53555/AJBR.v27i4S.4345.

[20] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, 2020, doi: 10.1186/s12864-019-6413-7.

[21] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PLoS One*, vol. 10, no. 3, p. e0118432, Mar. 2015, doi: 10.1371/journal.pone.0118432.

[22] S. Kruschel, N. Hambauer, S. Weinzierl, S. Zilker, M. Kraus, and P. Zschech, "Challenging the Performance-Interpretability Trade-Off: An Evaluation of Interpretable Machine Learning Models," *Business & Information Systems Engineering*, vol. 68, no. 1, pp. 159–183, Feb. 2026, doi: 10.1007/s12599-024-00922-2.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[23] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2016, pp. 1135–1144. doi: 10.1145/2939672.2939778.

[24] A. Barredo Arrieta *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020, doi: 10.1016/j.inffus.2019.12.012.

[25] A. Pareja *et al.*, "EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5363–5370, Apr. 2020, doi: 10.1609/aaai.v34i04.5984.

[26] Kusnawi, M. A. Wibowo, and Ridwan Sanjaya, "Real-Time Explainable Concept Drift Detection for Eco-Driving in Mining Trucks using KSWIN and Event-Triggered SHAP," *Journal of Information Systems and Informatics*, vol. 8, no. 2, pp. 1534–1556, Apr. 2026, doi: 10.63158/journalisi.v8i2.1551.