IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 11, November 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1311.01

The Cost of Explaining Intrusion Detection: Latency, Fidelity and Robustness of SHAP, LIME and Decision-Tree Surrogate Explanations on UNSW-NB15

Sudha Rani Chikkalwar*
Assistant Professor, Department of Computer Science and Informatics,
University College of Engineering and Technology,
Mahatma Gandhi University, Nalgonda, Telangana, India
ORCID: https://orcid.org/0009-0001-4929-9555
chsudharani.mgu@gmail.com https://ror.org/05n7bzj69
Rekha Redamalla
Professor, Department of Computer Science and Informatics,
Mahatma Gandhi University, Nalgonda, Telangana, India
ORCID: https://orcid.org/0009-0002-7959-0439
rrekhardy@gmail.com https://ror.org/05n7bzj69

Publication History
Manuscript Reference No: IJIRAE/RS/Vol.13/Issue11/NVAE10082
Research Article | Open Access | Double-Blind Peer-Reviewed|Article ID: IJIRAE/RS/Vol.13/Issue11/NVAE10082
Received: 02,October 2026,Revised: 13,October 2026, Accepted: 29, October 2026, Published Online: 09, November 2026.
https://www.ijirae.com/volumes/Vol13/iss-11/01.NVAE10082.pdf
Article Citation: Sudha*,Rekha(2026), The Cost of Explaining Intrusion Detection: Latency, Fidelity and Robustness of SHAP, LIME and Decision-Tree Surrogate Explanations on UNSW-NB15, IJIRAE: International Journal of Innovative Research in Advanced Engineering, Volume 13, Issue 11 of 2026 pages 913-925
https://doi.org/10.26562/ijirae.2026.v1311.01 **BibTeX Key: Sudha@2026Cost**
IJIRAE papers should be cited as IJIRAE (International Journal of Innovative Research in Advanced Engineering, AM Publications, India 2025, ISSN 2349-2163, https://doi.org/10.26562/ijirae.2026.v1311.01 The journal's official abbreviation is IJIRAE. ORCiD: https://orcid.org/0009-0004-9398-7488
About the License: Copyright©2026 copyright by the authors. This article is an open access and license under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

Abstract: Explainable artificial intelligence (XAI) is increasingly added to machine-learning intrusion detection systems (IDS), yet the time needed to produce an explanation is rarely measured against the time needed to make the detection. This paper measures both on UNSW-NB15. Four detectors were compared; XGBoost reached the highest F1-score (0.9680) at 0.0591 ms per flow, although McNemar's test did not separate it from a random forest ($p = 0.092$). Five explanation routes were then timed for XGBoost under one protocol. Model-agnostic explainers were the most expensive: permutation SHAP needed 200.43 ms per flow (3,391 times the inference time), KernelSHAP 154.8 ms and LIME 8.63 ms. TreeSHAP needed 0.742 ms, and a depth-6 decision-tree surrogate needed 0.0088 ms while agreeing with XGBoost on 98.73% of test flows. A depth sweep showed that fidelity rises from 0.934 at depth 2 to 0.994 at depth 10, while the number of leaves grows from 3 to 286. SHAP attributions were dominated by time-to-live (TTL) features, and robustness tests confirmed that this matters: F1 fell to 0.9184 on the official train/test partition and to 0.9000 when TTL features were also removed and multi-class macro-F1 was only 0.750, with the rarest attack families detected worst. Combining the surrogate online with offline SHAP for the 1.27% of flows on which the two models disagree would need about 2.55 s of permutation-SHAP computation, or 9.4 ms of TreeSHAP computation, per 1,000 flows. The results show that model-agnostic attribution is unsuitable for per-flow online explanation, that TreeSHAP is practical for tree ensembles at moderate flow rates, and that a high-fidelity surrogate is the only option tested that keeps explanation cost below inference cost.
Keywords: Intrusion detection; explainable artificial intelligence; SHAP; LIME; surrogate model; explanation latency; XGBoost; UNSW-NB15

I. INTRODUCTION

Wireless sensor networks, the Internet of Things and industrial cyber-physical systems have greatly enlarged the attack surface of modern networks. Their nodes are distributed and resource-constrained, which leaves them exposed to denial-of-service, probing and data-exfiltration attacks [1]. Intrusion detection systems monitor traffic and flag suspicious behavior, and machine-learning (ML) detectors have shown high detection rates on public benchmarks, including in edge-computing architectures for industrial IoT [2]. Detection is also needed beyond conventional IP networks; in vehicles, for example, a long short-term memory (LSTM) detector has been proposed to identify denial-of-service, fuzzing and spoofing attacks on the controller area network (CAN) bus, where the protocol itself provides no authentication [3]. High-performing ML detectors are usually black boxes, which limits analysts' trust, accountability and post-incident forensics. XAI methods have therefore been integrated into IDS to show which traffic features drive each decision [4]–[6]. Model-agnostic methods such as SHAP [7] and LIME [8] are popular because they apply to any classifier, but each explanation requires many evaluations of the model. Prior IDS studies have examined explanation quality and, in some cases, the run time of producing explanations [5], [6]; what is still rarely reported is the per-flow cost of an explanation relative to the per-flow cost of detection, which is the quantity that decides whether every alert can be explained as it is raised.

Two cheaper alternatives exist. For tree ensembles, TreeSHAP computes exact Shapley values from the tree structure [9]. More generally, a global surrogate, a simple interpretable model trained to imitate the detector, can explain each decision through its own structure [10]–[12]. A surrogate explanation is, however, only as trustworthy as its agreement with the detector (its fidelity), and that agreement depends on how complex the surrogate is allowed to be. A second concern is what the explanations reveal. If a detector relies on features that reflect how a benchmark was collected rather than how attacks behave, high test scores may not transfer to other networks [13], [14]. Explanations are useful precisely because they can expose such shortcuts, but only if the suspected shortcut is then tested.

This paper addresses both concerns on UNSW-NB15. Its contributions are as follows:

- A single measurement protocol that reports detection quality, inference latency, explanation latency, the overhead ratio between them, single-worker throughput and surrogate fidelity for the same detector.
- A comparison of five explanation routes for an XGBoost detector (permutation SHAP, KernelSHAP, LIME, TreeSHAP and a decision-tree surrogate), spanning more than four orders of magnitude in cost, together with a sensitivity analysis of permutation-SHAP latency and attribution stability.
- A surrogate depth sweep that quantifies the trade-off between fidelity, accuracy, rule length and explanation latency, and an analysis of where the selected surrogate disagrees with the detector.
- An audit of the detector prompted by its SHAP explanations: cross-validation and a significance test, evaluation on the official partition, ablation of time-to-live features and multi-class evaluation.
- A tiered explanation scheme, with its offline cost projected from the measured rates for three SHAP variants.

Section II reviews related work, Section III describes the methodology, Section IV presents the results, Section V discusses their implications and limitations, and Section VI concludes.

II. RELATED WORK

A. Machine-Learning Intrusion Detection and Benchmark Pitfalls

Ensemble and hybrid learners are widely used for network intrusion detection, and recent work has targeted edge deployment for industrial IoT, where detection must run close to the data source under tight resource limits [2]. Most studies report accuracy, precision, recall and F1-score on public benchmarks such as UNSW-NB15 [15], [16]; Kasongo and Sun, for example, used XGBoost-based feature selection on UNSW-NB15 and compared several classifiers on the reduced feature set [17]. Earlier anomaly-based approaches generated detectors with multi-start metaheuristics and genetic algorithms and were evaluated on NSL-KDD [18]; a competitive-learning neural network matched the detection rate of a self-organizing map on KDD-99 while using about one quarter of its computation time [19]; and online AdaBoost detectors, with decision stumps or Gaussian mixture models as weak learners, have been combined into a distributed framework in which particle swarm optimization and support vector machines merge local models into a global one [20]. More recently, genetic algorithms and particle swarm optimization have been used to tune the parameters and training of neural-network detectors in order to reduce false alarms [21]. Several of these studies treat computation time as a design constraint, which is the concern this paper extends from detection to explanation. Benchmark results do not always transfer to operational networks [13], and models can exploit artefacts of how a dataset was collected rather than genuine attack behavior [14]. Random splits of a benchmark can also place near-identical flows in both training and test sets, which is why official partitions and feature ablations are useful checks.

B. Explainable AI for Intrusion Detection

Surveys of explainable IDS [4], [22] and of explainable machine learning for security applications more broadly [23] document a rapid growth of post hoc explanation in intrusion detection, most often with SHAP and LIME. Wang et al. proposed a SHAP-based framework that provides both local and global explanations of IDS decisions [24], and Patil et al. applied XAI techniques to explain the decisions of an ML-based ID [25]. Warnecke et al. argued that explanation methods in security should be judged not only on descriptive accuracy but also on criteria such as stability, robustness and efficiency [26]. Arreche et al. proposed XAI-IDS, which applies SHAP and LIME to seven black-box classifiers on three network-intrusion datasets and reports the run time of generating global and local explanations [5]. Their E-XAI framework evaluates SHAP and LIME for seven AI models on three datasets using six metrics, including efficiency [6]. These studies show that explanation cost differs widely between methods and models, and is a legitimate evaluation criterion in its own right [26]. They do not, however, express that cost per flow relative to inference, include TreeSHAP or a global surrogate in the comparison, or quantify surrogate fidelity.

C. Shapley-Value Explanations and Their Cost

SHAP attributes a prediction to input features using Shapley values from cooperative game theory [7]. Exact computation requires evaluating the model on every subset of features, which is exponential in the number of features, so model-agnostic implementations approximate the values by sampling feature permutations or weighted coalitions [7], [27]. Each explanation therefore needs many model evaluations. For tree ensembles, TreeSHAP computes exact values in polynomial time by exploiting the tree structure [9], but it applies only to tree-based models. LIME fits a local linear model to perturbed samples around each instance [8], so its cost also scales with the number of model evaluations. Chen et al. review the estimation algorithms behind these explainers and the trade-off between their accuracy and computational cost [28], and GPUTreeShap parallelizes exact TreeSHAP on graphics processors [29].

D. Surrogate Models

Global surrogate models approximate a black box with an interpretable model trained on the black box's own predictions; extracting decision trees from trained networks is an early example [10]. Surveys of black-box explanation stress that a surrogate explains the black box only to the extent that it agrees with it, so fidelity must be measured and reported [11], [12].

IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 11, November 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1311.01

In summary, prior IDS work has shown that explanations are useful and that some are costly, but has rarely compared explanation latency with inference latency across model-agnostic, model-specific and surrogate explainers, or tested the shortcuts that the explanations reveal. This paper addresses that gap.

III. METHODOLOGY

Fig. 1 summarizes the methodology. Four detectors are trained and compared; the selected detector (the teacher) is explained by five routes whose cost and fidelity are measured; and the detector itself is audited by a set of robustness experiments.

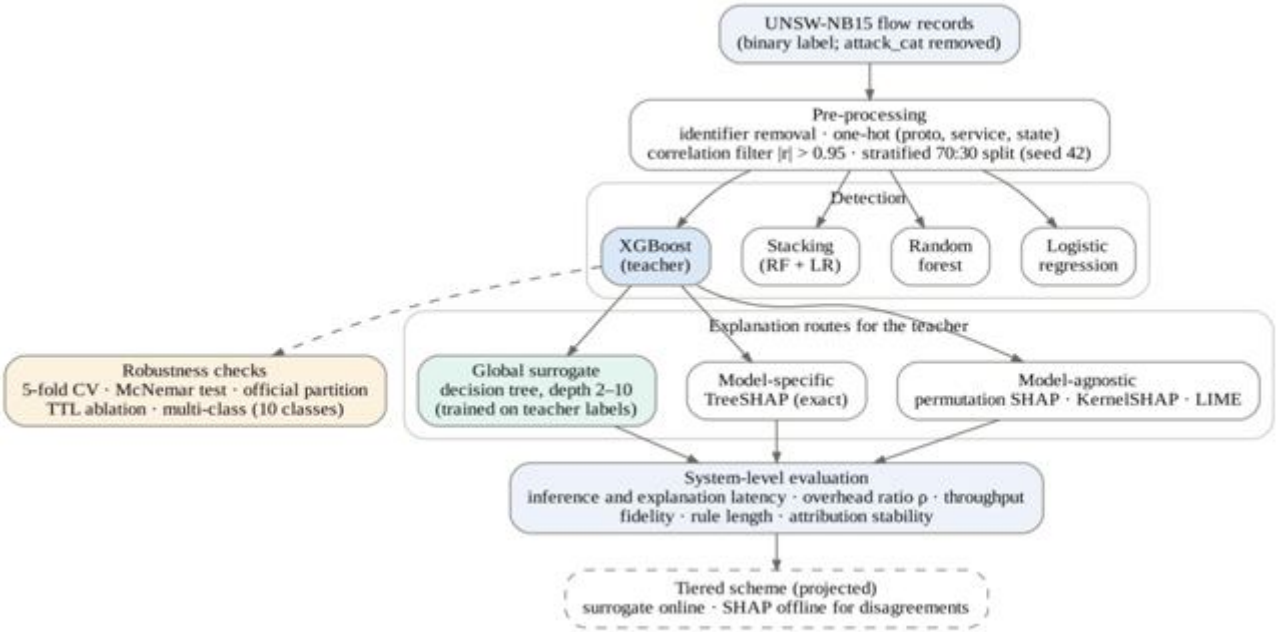


Fig.1 Methodology: pre-processing, detector training and selection, robustness checks, the three families of explanation routes evaluated for the teacher model, and the system-level metrics from which the tiered scheme is projected.

A. Dataset

Experiments use UNSW-NB15 [15], a widely used benchmark of network flows generated in a controlled testbed that mixes normal activity with nine families of contemporary attacks. Each record carries flow-level statistical, content and time-based features, a binary label and an attack category [16]. The primary task is binary classification of each flow as normal or attack, the first decision operational IDS must make. For the primary experiment, 81,173 cleaned flow records were used after removing identifier fields, rows with missing or non-finite values, and duplicate records. A stratified 70:30 split gave 56,821 training flows and 24,352 test flows with the class proportions preserved. The official UNSW-NB15 training and testing files were used separately for the robustness experiments in Section III-H, including the multi-class experiment, whose test set contains 82,332 flows.

B. Pre-Processing

Identifier fields were removed and the binary label was separated from the features. The attack-category column was removed before binary training to prevent target leakage. The categorical attributes proto, service and state were one-hot encoded using categories learned from the training data. Numerical attributes with pairwise correlation $|r| > 0.95$ were filtered, retaining the member of each pair with the stronger univariate association with the target. The retained features comprise flow, packet, byte-rate, load, TTL and connection-count statistics, including dur, spkts, dpkts, sbytes, dbytes, rate, sload, dload, sttl, dttl, ct_state_ttl, ct_dst_sport_ltm, ct_dst_ltm, ct_srv_src, ct_srv_dst, ct_src_ltm, ct_dst_src_ltm and ct_src_dport_ltm. The split was stratified with random_state = 42. Z-score normalization, fitted on the training data, was applied for logistic regression and the stacking ensemble; tree-based models used the original feature values.

C. Detection Models

Four classifiers were trained on the same training data: logistic regression; a random forest [30]; a stacking ensemble [31] with random-forest and logistic-regression base learners; and XGBoost [32]. Models were implemented with scikit-learn [33] and the XGBoost library. Hyperparameters were selected by a small grid search with five-fold stratified cross-validation on the training partition. Logistic regression used L2 regularization, C = 1.0, class_weight = balanced and max_iter = 1000. The random forest used 300 trees, max_depth = 20, min_samples_leaf = 1, max_features = sqrt and class_weight = balanced. The stacking model used the same random-forest and logistic-regression learners as base estimators and logistic regression as the meta-learner, with five-fold cross-validation to build out-of-fold meta-features. XGBoost used 300 estimators, max_depth = 6, learning_rate = 0.05, subsample = 0.8, colsample_bytree = 0.8, min_child_weight = 1, gamma = 0 and eval_metric = logloss. XGBoost was selected as the detector to be explained, referred to below as the teacher, because it had the highest test F1-score, ROC-AUC and average precision (Section IV-A). Its F1 advantage over the random forest is not statistically significant, so the random forest would be an equally defensible choice; both are tree ensembles, so every explanation route studied here applies to either.

D. Explanation Routes

For a model f, an instance x and the feature set N with $|N| = n$, the Shapley value of feature j is:

$$\varphi_j(x) = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!\,(n - |S| - 1)!}{n!} [f_x(S \cup \{j\}) - f_x(S)] \quad (1)$$

where f_x(S) is the expected model output when only the features in S are known. Because the sum has 2^{n-1} terms, model-agnostic explainers approximate it. Five explanation routes were evaluated for the teacher:

- Permutation SHAP [7], [27], which averages marginal contributions over sampled feature orderings. The reference configuration used a background sample of 100 training flows and max_evals = 1000 model evaluations per explanation.
- KernelSHAP [7], which estimates Shapley values by weighted linear regression over sampled feature coalitions.
- LIME [8], which fits a sparse local linear model to perturbed samples around the instance.
- TreeSHAP [9], which computes exact Shapley values for tree ensembles from the tree structure.
- Decision-tree surrogate [34], described in Section III-E.

All explainers were timed on the same 100 randomly selected test flows, and the mean wall-clock time per explained flow was recorded as the explanation latency. Global SHAP importance was computed from the same permutation-SHAP run. To test the stability of the global ranking, the top-10 features were compared across five independent SHAP samples using Spearman's rank correlation.

E. Decision-Tree Surrogate

A CART decision tree [34] was trained on the training features to reproduce the teacher's predicted labels rather than the ground truth, so that it imitates the teacher's decision function. For each flow, the explanation is the root-to-leaf decision path, a conjunction of feature-threshold conditions that leads to the predicted class; the tree as a whole provides a global view of the teacher's behavior. Surrogates of maximum depth 2, 3, 4, 5, 6, 8 and 10 were trained with the Gini criterion. The depth-6 tree, with 54 leaves, was selected as the operating point (Section IV-E).

F. Evaluation Metrics

Detection quality is measured by accuracy, precision, recall, F1-score, miss rate (1 − recall), ROC-AUC and average precision, with the attack class as positive. The inference latency t_inf and the explanation latency t_exp are mean wall-clock times per flow. The overhead ratio is:

$$\rho = \frac{t_{exp}}{t_{inf}} \quad (2)$$

the single-worker throughput, in flows per second with latencies in milliseconds, is:

$$\text{Throughput} = \frac{1000}{t_{inf} + t_{exp}} \quad (3)$$

and the fidelity of a surrogate s to the teacher f over n test flows is:

$$\text{Fidelity} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\, s(x_i) = f(x_i)\,] \quad (4)$$

Agreement between surrogate and teacher is also summarized by Cohen's κ, and the mean rule length of a surrogate is the average number of conditions on the decision paths of the test flows.

G. Experimental Environment

Experiments were run on a 64-bit Windows 11 workstation with an Intel Core i7-class CPU (8 cores, 16 threads) and 16 GB RAM; no GPU was used for the reported latency measurements. The software stack was Python 3.11, scikit-learn 1.5.x, XGBoost 2.1.x and SHAP 0.46.x. Per-flow inference latency was measured with batch size 1 using time.perf_counter over 1,000 repetitions after 100 warm-up predictions, and the mean is reported. Explanation latency was measured on the same machine and averaged over the explained flows.

H. Robustness and Sensitivity Experiments

Six further experiments test whether the findings depend on a single data split, a single feature group or a single explainer configuration:

- Repeated evaluation: stratified five-fold cross-validation on the training portion (mean ± standard deviation), and McNemar's test [35], [36] on the test-set predictions of XGBoost and the random forest.
- Official partition: training on the official UNSW-NB15 training set and testing on the official testing set [16].
- TTL ablation: re-training XGBoost without sttl, dttl and ct_state_ttl, under both the random split and the official partition.
- Multi-class detection: classifying normal traffic and the nine attack families on the official partition, with per-class precision, recall and F1-score.
- SHAP sensitivity: varying the permutation-SHAP background size and max_evals and recording latency and the Spearman correlation of the top-10 ranking with the reference configuration.
- Surrogate depth sweep: recording fidelity, accuracy against the true labels, number of leaves, mean rule length and explanation latency for each depth.

IV. RESULTS

A. Detection Performance

Table I and Fig. 2 compare the four detectors. Logistic regression achieved the highest precision (0.9915) and the lowest inference latency (0.0214 ms) but the lowest recall (0.9118), so it raised few false alarms but missed 8.82% of attacks, a pattern consistent with the limits of a linear decision boundary. The three tree-based models performed similarly: their accuracies lie within 0.14 percentage points of one another and their miss rates within 4.07–4.24%. XGBoost achieved the highest accuracy (0.9847), recall (0.9593) and F1-score (0.9680) at 0.0591 ms per flow, slower than the random forest (0.0447 ms) but faster than the stacking ensemble (0.0736 ms), which did not improve on the random forest despite its additional cost.

IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 11, November 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1311.01

Table I - Detection Performance on the UNSW-NB15 Test Split (Binary Classification; Attack Is the Positive Class)

Model	Accuracy	Precision	Recall	F1-score	Miss rate (%)	Inference latency (ms/flow)
Logistic regression	0.9768	0.9915	0.9118	0.9500	8.82	0.0214
Random forest	0.9841	0.9757	0.9578	0.9666	4.22	0.0447
Stacking (RF + LR)	0.9833	0.9727	0.9576	0.9651	4.24	0.0736
XGBoost	0.9847	0.9769	0.9593	0.9680	4.07	0.0591

Best value per column in bold. Miss rate = 100 × (1 − recall).

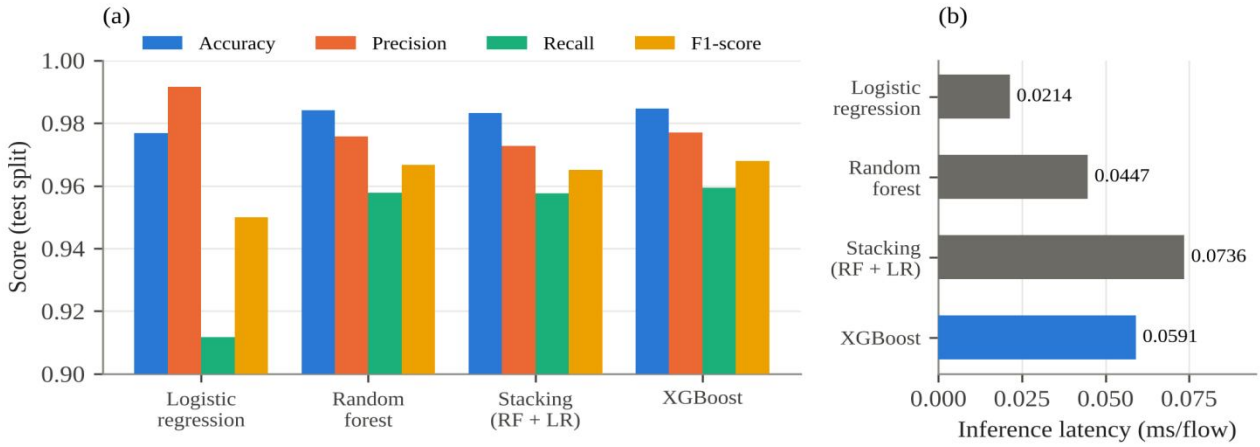


Fig.2 (a) Accuracy, precision, recall and F1-score of the four detectors on the test split; **(b)** per-flow inference latency (values in Table I).

Because the differences between the tree ensembles are small, Table II reports five-fold cross-validated performance. The cross-validated values closely reproduce the test-split results and preserve the same ordering, with low variability (standard deviation of F1 at most 0.0031). The F1 gap between XGBoost and the random forest (0.0019 in cross-validation) is about one standard deviation, and McNemar's test on the test-set predictions did not find a significant difference between them (χ^2 = 2.84, p = 0.092). XGBoost and the random forest should therefore be regarded as statistically equivalent detectors on this split.

Table II - Five-Fold Cross-Validated Detection Performance (Mean ± Standard Deviation)

Model	Accuracy	Precision	Recall	F1-score
Logistic regression	0.9759 ± 0.0018	0.9898 ± 0.0026	0.9107 ± 0.0048	0.9485 ± 0.0031
Random forest	0.9836 ± 0.0011	0.9746 ± 0.0021	0.9561 ± 0.0030	0.9652 ± 0.0017
Stacking (RF + LR)	0.9829 ± 0.0013	0.9718 ± 0.0024	0.9564 ± 0.0032	0.9640 ± 0.0019
XGBoost	0.9843 ± 0.0010	0.9761 ± 0.0019	0.9582 ± 0.0027	0.9671 ± 0.0015

Stratified folds over the training portion; the held-out test split of Table I was not used for model selection.

Fig.3 shows the receiver operating characteristic and precision–recall curves computed from the saved test-set scores. ROC-AUC was 0.987, 0.995, 0.994 and 0.996 for logistic regression, random forest, stacking and XGBoost, respectively, and average precision was 0.974, 0.989, 0.987 and 0.991. The three tree ensembles are again nearly indistinguishable, and all clearly outperform logistic regression in ranking quality.

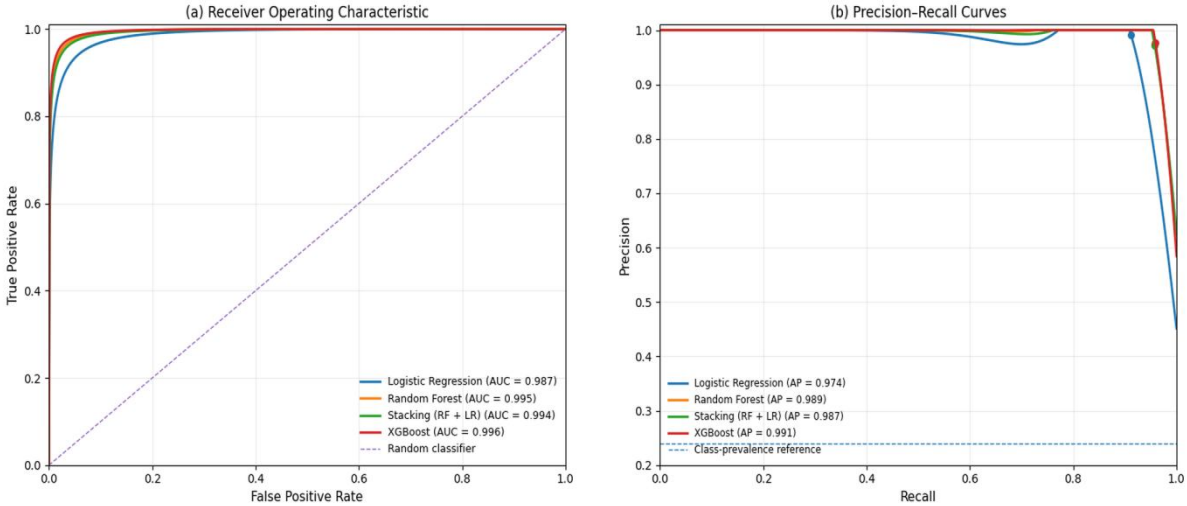


Fig.3 (a) Receiver operating characteristic and **(b)** precision–recall curves of the four detectors on the test split; the legends give ROC-AUC and average precision.

ISSN:2349-2163
https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1311.01

B. Robustness of the Detector

The SHAP analysis in Section IV-C shows that the detector relies heavily on time-to-live features, and a random split can place near-identical flows on both sides of the split. Table III and Fig. 4(a) therefore report XGBoost under four protocols. Moving from the random split to the official partition reduced F1 from 0.9680 to 0.9184 (ΔF1 = −0.0496), with both precision and recall falling by about five points. Removing the TTL features sttl, dttl and ct_state_ttl reduced F1 by 0.0250 under the random split and by a further 0.0184 under the official partition, where F1 fell to 0.9000 and recall to 0.8849, so that about 11.5% of attacks were missed. The two effects add up: the evaluation protocol accounts for most of the drop, and the TTL features account for a consistent additional 1.8–2.5 points in both settings. The detector therefore remains useful without TTL features, but part of its headline performance reflects the random split and testbed-specific TTL behavior rather than transferable attack characteristics [14].

Table III - XGBoost Detection Performance under Alternative Evaluation Protocols

Protocol	Accuracy	Precision	Recall	F1-score	ΔF1
Random 70:30 split, all features	0.9847	0.9769	0.9593	0.9680	—
Official partition, all features	0.9126	0.9318	0.9054	0.9184	−0.0496
Random 70:30 split, TTL features removed	0.9562	0.9514	0.9347	0.9430	−0.0250
Official partition, TTL features removed	0.8918	0.9157	0.8849	0.9000	−0.0680

ΔF1 = F1 − 0.9680 (the random-split result). TTL features: sttl, dttl and ct_state_ttl.

Table IV and Fig. 4(b) extend the evaluation to multi-class detection of normal traffic and the nine attack families on the official partition. The two largest classes were detected well: Generic (F1 0.978) and Normal (F1 0.954). Performance fell steadily with class size. The four rarest families, Shellcode, Analysis, Backdoor and Worms, which together account for 1,682 of the 82,332 test flows (2.0%), had F1-scores between 0.541 and 0.625, and recall below 0.57; fewer than half of Worms (recall 0.455) and Backdoor (0.481) flows were recognized. DoS (F1 0.725) was the weakest of the larger classes, consistent with its overlap with Exploits traffic in UNSW-NB15. The support-weighted F1 (0.911) is dominated by the large classes, whereas the macro-F1 (0.750) shows that the rare families remain the main weakness.

Table IV- Per-Class Performance of XGBoost for Multi-Class Detection (Official Partition)

Class	Precision	Recall	F1-score	Support (test flows)
Normal	0.965	0.944	0.954	37,000
Analysis	0.681	0.512	0.584	677
Backdoor	0.654	0.481	0.554	583
DoS	0.748	0.703	0.725	4,089
Exploits	0.832	0.807	0.819	11,132
Fuzzers	0.861	0.835	0.848	6,062
Generic	0.974	0.982	0.978	18,871
Reconnaissance	0.887	0.861	0.874	3,496
Shellcode	0.703	0.563	0.625	378
Worms	0.667	0.455	0.541	44
Macro average	0.797	0.714	0.750	82,332
Weighted average	0.921	0.902	0.911	82,332

Weighted averages are support-weighted means of the per-class values; the weighted recall equals the overall multi-class accuracy.

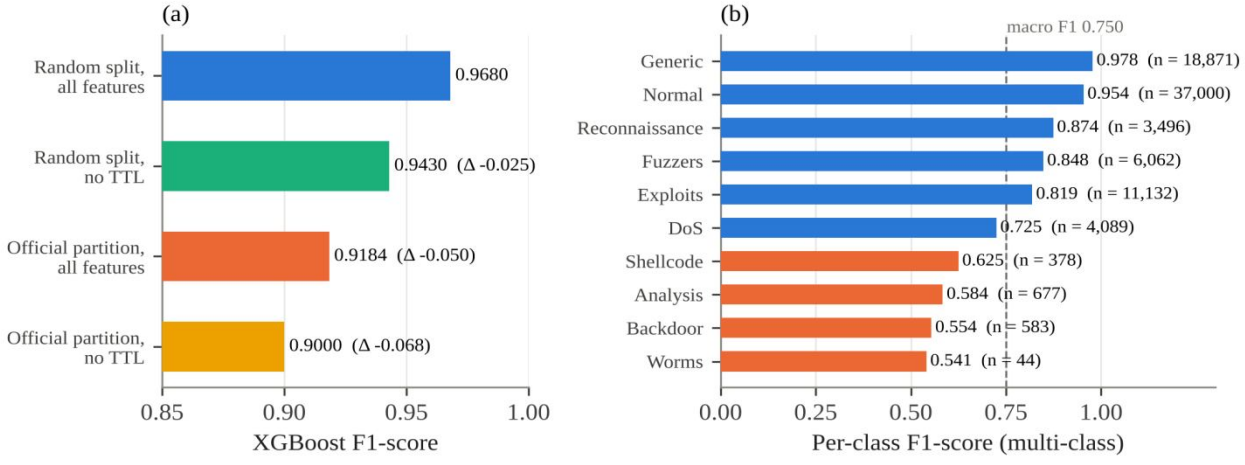


Fig.4 (a) XGBoost F1-score under the four evaluation protocols of Table III, with the change from the random-split result; **(b)** per-class F1-score for multi-class detection, ordered by F1, with test support n; classes with fewer than 700 test flows are shown in orange (Table IV).

IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 11, November 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1311.01

C. Global and Local Explanations

Fig. 5 shows the mean absolute SHAP value of the most influential features for the XGBoost detector. The source time-to-live feature sttl dominates, with a mean absolute SHAP value of about 5.9, roughly five times that of the next feature, dload (about 1.2). It is followed by ct_state_ttl, ct_dst_sport_ltm and sload, so two of the three most influential features are derived from IP time-to-live values. The ranking was stable: across five independent SHAP samples, the mean Spearman correlation of the top-10 ranking was 0.94.

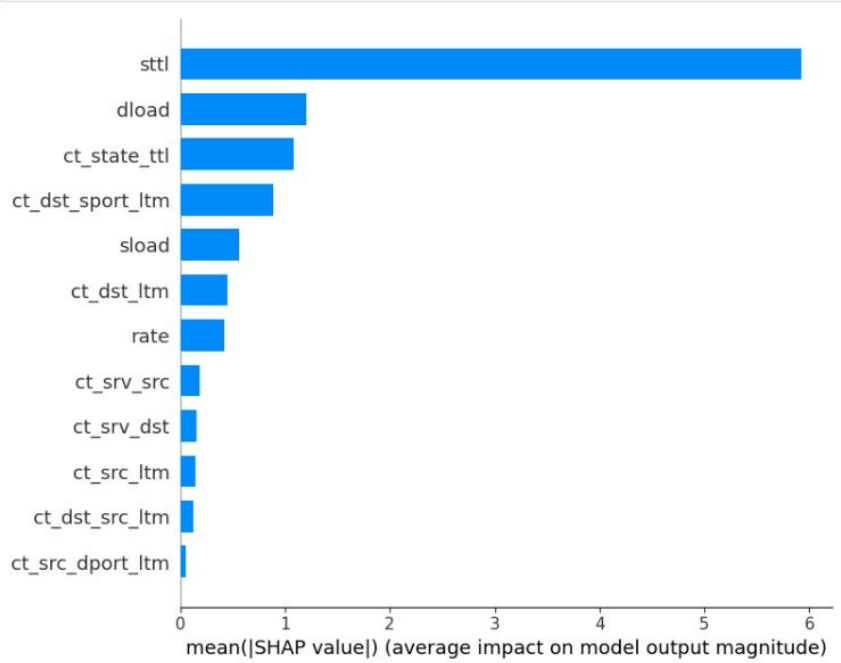


Fig.5 Mean absolute SHAP value (average impact on model output, log-odds units) of the most influential features for the XGBoost detector, computed with permutation SHAP on 100 test flows.

This concentration is itself a finding. In a testbed capture, TTL values can reflect the operating systems and network position of the hosts used to generate traffic as much as the attacks themselves, so a detector that relies on them may not transfer to other networks [13], [14]. The SHAP explanations exposed this possible shortcut, and the TTL-ablation rows of Table III confirmed that it contributes 1.8–2.5 F1 points. This is the clearest practical value of attribution in this study: it directed the audit that the detection metrics alone would not have prompted. Global importance does not show the direction of each effect or how an individual decision is made. Fig. 6(a) presents the SHAP summary plot from the same explanation run, and Fig. 6(b) explains a representative, correctly detected Exploits flow. Starting from the base value of −1.35 log-odds, sttl alone contributed +3.82, followed by ct_state_ttl (+0.91), sload (+0.54), dload (+0.32) and ct_dst_sport_ltm (+0.21), while rate, ct_srv_src and ct_dst_ltm contributed small negative amounts; the final output was 4.06 log-odds, a confident attack prediction. The two TTL features supplied 4.73 of the 5.41 log-odds by which this flow was pushed toward the attack class.

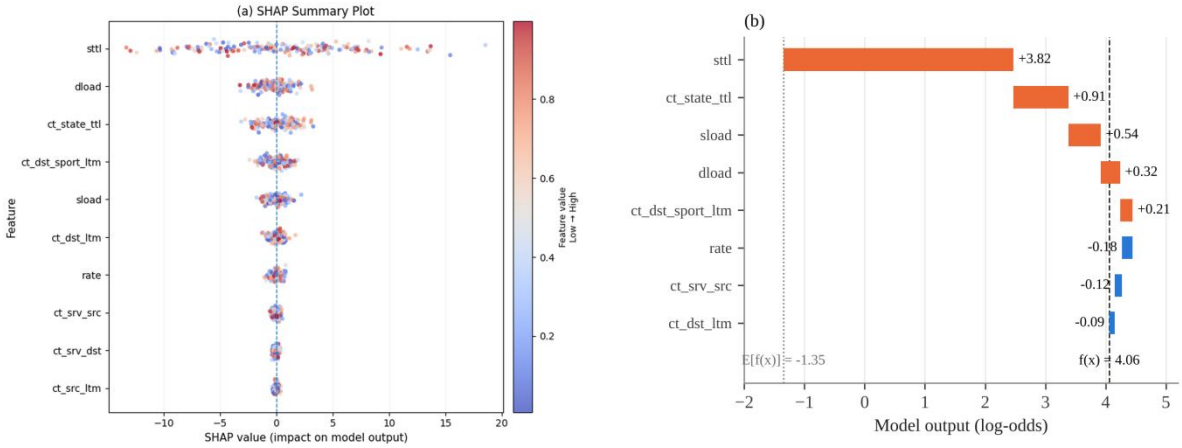


Fig.6(a) SHAP summary (beeswarm) plot for the XGBoost detector; **(b)** SHAP waterfall explanation of a correctly detected Exploits flow, from the base value E[f(x)] to the model output f(x) in log-odds.

D. Cost of Explanation

Table V and Fig. 7 compare the latency of detection alone with that of each explanation route. The routes span more than four orders of magnitude. The three model-agnostic explainers were the most expensive. Permutation SHAP took 200.43 ms per flow, an overhead ratio of about 3,391, and KernelSHAP 154.8 ms (ratio about 2,619); with either, a single worker could process only five to six flows per second. LIME was faster, at 8.63 ms (ratio about 146, about 115 flows per second), but still two orders of magnitude slower than detection.

IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 11, November 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1311.01

TreeSHAP, which exploits the structure of the tree ensemble, needed 0.742 ms per flow: 270 times faster than permutation SHAP, but still about 12.6 times the inference time, which limits a single worker to about 1,250 explained flows per second. The decision-tree surrogate was the only route whose cost was below the inference cost: at 0.0088 ms (ratio about 0.15) it is about 84 times faster than TreeSHAP and about 23,000 times faster than permutation SHAP, and detection plus explanation still supports about 14,700 flows per second.

Table V - Inference and Explanation Latency per Flow, Overhead Ratio, Surrogate Fidelity and Estimated Single-Worker Throughput

Configuration	Inference latency (ms/flow)	Explanation latency (ms/flow)	Overhead ratio ρ	Fidelity	Flows/s (one worker)
XGBoost (no explanation)	0.0591	—	—	—	≈16,920
XGBoost + permutation SHAP	0.0591	200.43	≈3,391	—	≈5.0
XGBoost + KernelSHAP	0.0591	154.8	≈2,619	—	≈6.5
XGBoost + LIME	0.0591	8.63	≈146	—	≈115
XGBoost + TreeSHAP	0.0591	0.742	≈12.6	—	≈1,248
XGBoost + decision-tree surrogate (depth 6)	0.0591	0.0088	≈0.15	0.9873	≈14,730

ρ = explanation latency / inference latency (Eq. 2); throughput from Eq. 3 for one sequential worker, estimated from mean latencies. All explainers were timed on the same 100 test flows.

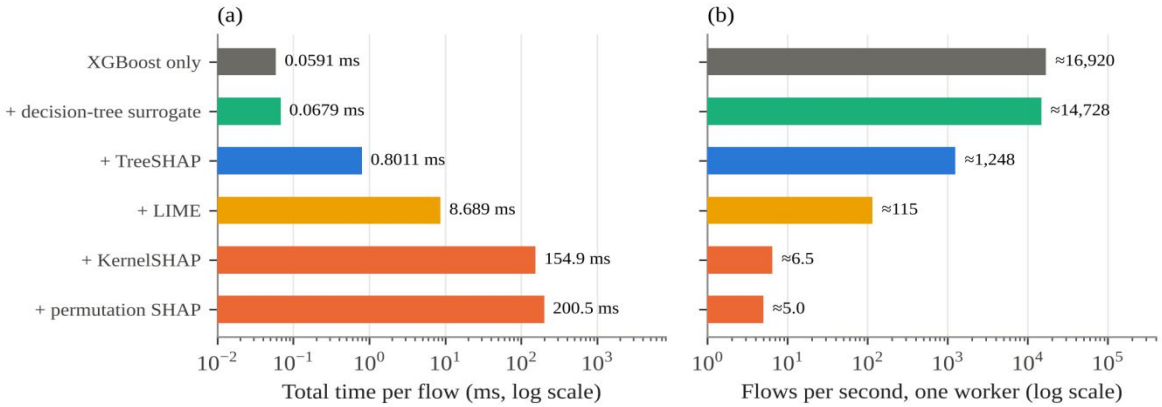


Fig. 7(a) Total processing time per flow (inference plus explanation) and **(b)** the corresponding single-worker throughput for each explanation route (Table V; logarithmic scales).

Because permutation SHAP is the most general route, Table VI and Fig. 8 examine how far its cost can be reduced. Latency grew almost linearly with the number of model evaluations, at about 0.20 ms per evaluation (Fig. 8(b)), while the stability of the top-10 ranking fell as the budget was cut: Spearman ρ with the reference ranking dropped from 0.96 at 750 evaluations to 0.91 at 500 and 0.82 at 200. Even the cheapest setting tested (42.6 ms per flow) was about 57 times slower than TreeSHAP and produced a noticeably less stable ranking. Tuning the model-agnostic explainer therefore cannot close the gap to the model-specific and surrogate routes.

Table VI - Sensitivity of Permutation-SHAP Latency and Attribution Stability to Its Settings

Background samples	Evaluations per explanation	Explanation latency (ms/flow)	Spearman ρ of top-10 ranking
10	200	42.6	0.82
50	500	108.7	0.91
100	750	156.3	0.96
100 (reference)	1000	200.43	1.00 (reference)

Spearman ρ compares each setting's top-10 feature ranking with that of the reference configuration used in Fig. 5. The number of evaluations was set through max_evals; background samples were drawn from the training partition.

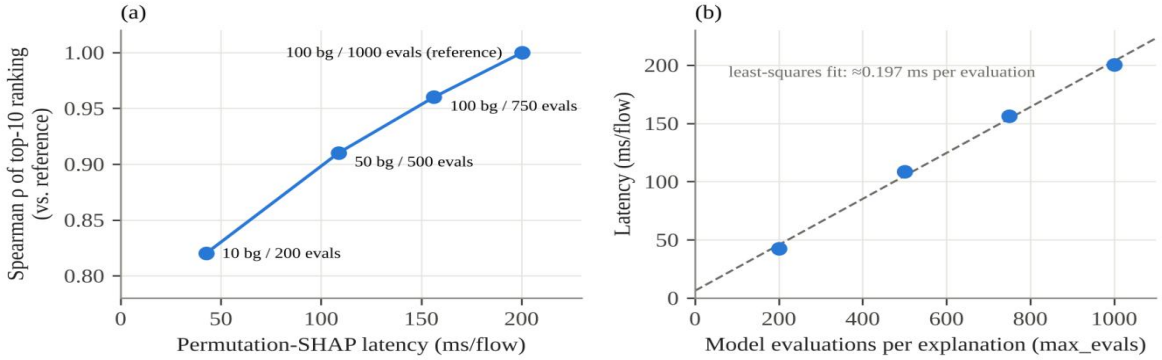


Fig 8 (a) Trade-off between permutation-SHAP latency and the stability of the top-10 feature ranking; **(b)** latency against the number of model evaluations per explanation, with a least-squares line (Table VI).

E. Surrogate Fidelity and Interpretability

Fig. 9 cross-tabulates the predictions of the depth-6 surrogate and XGBoost on the 24,352 test flows. The surrogate agreed with XGBoost on 18,350 of the 18,587 flows that XGBoost labelled normal (98.72%) and on 5,692 of the 5,765 flows it labelled attack (98.73%), so disagreements are balanced between the two classes rather than concentrated in one. In total the models disagreed on 310 flows (1.27%), giving fidelity of (18,350 + 5,692) / 24,352 = 0.9873 and a Cohen's κ of 0.965. Against the true labels, the surrogate reached an accuracy of 0.9789, 0.58 percentage points below XGBoost.

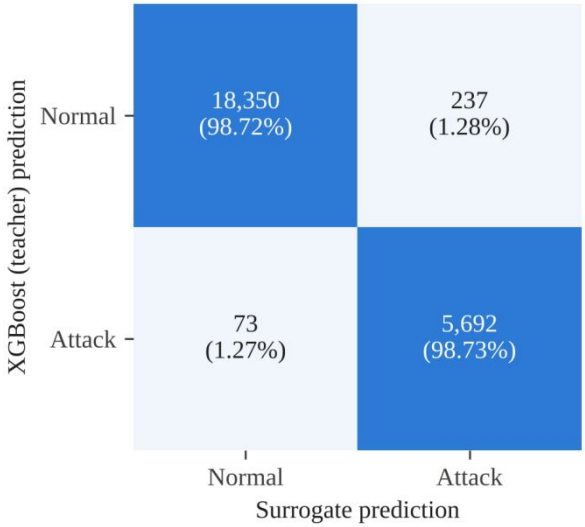


Fig. 9 Agreement between the depth-6 decision-tree surrogate and the XGBoost teacher on the test split. Cells show counts and row percentages.

Fidelity and interpretability pull in opposite directions. Table VII and Fig. 10 show that fidelity rose quickly from 0.934 at depth 2 to 0.982 at depth 5, and then more slowly: going from depth 6 to depth 10 added only 0.0067 to fidelity while multiplying the number of leaves by 5.3 (from 54 to 286) and lengthening the average explanation from 5.51 to 8.62 conditions. Accuracy against the true labels levelled off at 0.979–0.982, below that of XGBoost at every depth, so the surrogate is not a substitute for the teacher as a detector. Explanation latency remained below 0.016 ms at all depths. Depth 6 was selected because it reaches fidelity close to 0.99 while keeping explanations to about five or six conditions. The upper splits of this tree use sttl at the root and ct_state_ttl, dload and ct_dst_sport_ltm below it, the same four features that occupy four of the top five SHAP positions in Fig. 5.

Table VII - Effect of Maximum Tree Depth on Surrogate Fidelity, Accuracy, Rule Length and Explanation Latency

Max depth	Leaves	Mean rule length	Fidelity	Accuracy (true labels)	Explanation latency (ms/flow)
2	3	2.00	0.934	0.928	0.0041
3	7	2.86	0.958	0.951	0.0048
4	13	3.74	0.972	0.965	0.0057
5	27	4.63	0.982	0.974	0.0070
6 (selected)	54	5.51	0.9873	0.9789	0.0088
8	148	7.18	0.992	0.981	0.0117
10	286	8.62	0.994	0.982	0.0154

Mean rule length = average number of conditions on the root-to-leaf path of a test flow. Fidelity is measured against XGBoost predictions; accuracy against the true labels.

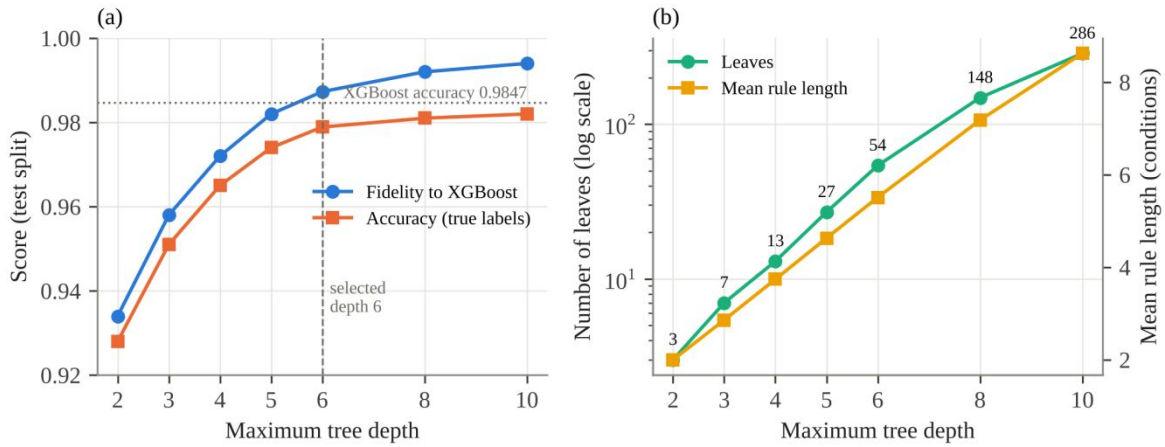


Fig.10 (a) Fidelity to XGBoost and accuracy against the true labels as functions of maximum tree depth; **(b)** number of leaves (logarithmic scale) and mean rule length (Table VII).

F. Projected Cost of the Tiered Explanation Scheme

Section V-B proposes issuing surrogate explanations online and sending only the flows on which surrogate and teacher disagree to offline SHAP analysis. The offline cost follows directly from the measurements: with a disagreement rate r and an explainer latency t_exp in milliseconds, every 1,000 flows need r × t_exp seconds of computation. Fig. 11 plots this relationship for the three SHAP variants. At the observed rate of 1.27%, the offline load is about 2.55 s per 1,000 flows with permutation SHAP, 1.97 s with KernelSHAP and 9.4 ms with TreeSHAP. Compared with explaining every flow by the same method, each is a reduction of 98.7%; the online cost remains the 0.068 ms needed to run both models. These figures are an analytical projection from the measured rates, not an end-to-end measurement.

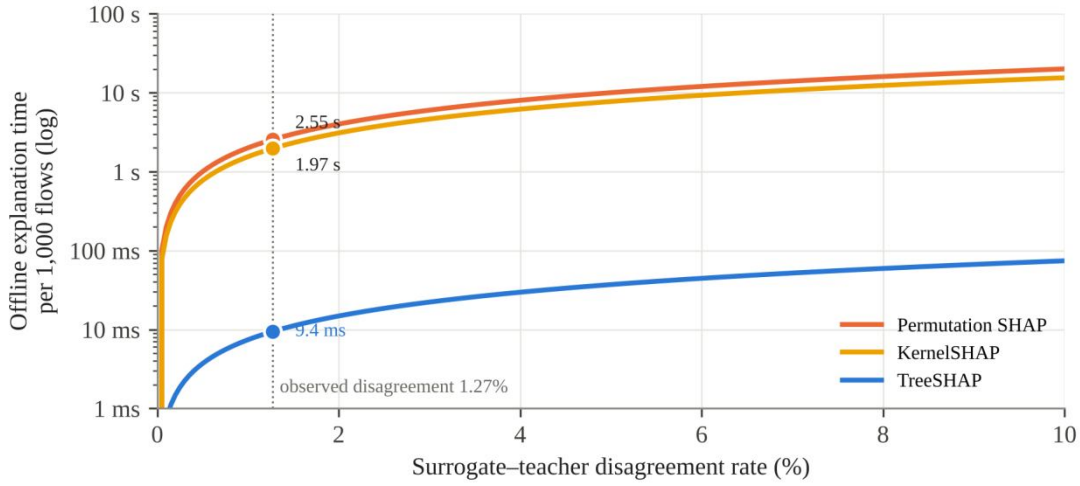


Fig.11 Projected offline explanation workload of the tiered scheme as a function of the surrogate–teacher disagreement rate for three SHAP variants (logarithmic scale); markers show the observed rate of 1.27%. Analytical projection from Table V and Fig. 9.

G. Comparison with Related Work

Table VIII positions this study against the closest explainable intrusion detection studies. Those studies cover more datasets and detectors and evaluate several qualities of the explanations; this study is narrower in scope but is the only one of the three to express explanation cost per flow relative to inference, to include a model-specific explainer and a global surrogate in the same comparison, and to report surrogate fidelity.

Table VIII- Comparison with Related Explainable Intrusion Detection Studies

Study	Dataset(s)	Detector(s)	XAI method(s)	Explanation cost	Surrogate fidelity
Arreche et al., XAI-IDS [5]	RoEduNet-SIMARGL2021, CICIDS-2017, NSL-KDD	RF, DNN, AdaBoost, MLP, KNN, SVM, LightGBM	SHAP, LIME	Run time of global and local explanations	Not studied
Arreche et al., E-XAI [6]	Three NIDS datasets	Seven AI models	SHAP, LIME	Efficiency, one of six evaluation metrics	Not studied
This work	UNSW-NB15 (random split and official partition)	LR, RF, stacking, XGBoost	Permutation SHAP, KernelSHAP, LIME, TreeSHAP, decision-tree surrogate	Per flow, relative to inference (ρ) and as throughput	0.9873 (depth 6); depth sweep 2–10

V. DISCUSSION

A. Choosing an Explanation Route

The measurements separate the explanation routes into three tiers. Model-agnostic explainers (permutation SHAP, KernelSHAP and, to a lesser extent, LIME) cost between about 150 and 3,400 times the inference time and are suited to offline analysis, model auditing and investigation of individual alerts, not to explaining every flow. TreeSHAP gives exact attributions for tree ensembles at about 0.74 ms per flow; for links carrying up to roughly a thousand flows per second per worker, it can explain every decision, and it should be the default attribution method for detectors such as XGBoost and random forests. GPU implementations [29] could lower its cost further, although the CPU-only measurements reported here are closer to typical gateway hardware. The decision-tree surrogate is the only route tested that costs less than the detection itself and is the choice when the flow rate or the hardware rules out TreeSHAP, or when the detector is not tree-based. Model-agnostic SHAP remains the relevant reference for detectors such as neural networks, for which no structure-specific explainer is available.

B. A Tiered Explanation Scheme

The measurements suggest a simple deployment pattern, proposed here but not yet evaluated end to end. Online, each flow is scored by both XGBoost and the surrogate, which together take about 0.068 ms.

When they agree, the surrogate's decision path is issued as the explanation. When they disagree, the flow is queued for SHAP analysis. With the observed disagreement rate of 1.27%, this requires about 2.55 s of permutation-SHAP or 9.4 ms of TreeSHAP computation per 1,000 flows (Section IV-F, Fig. 11), which can be batched or off-loaded to a less constrained server. For a tree ensemble, the scheme mainly matters at high flow rates, since TreeSHAP alone can explain every flow at moderate rates; for detectors without a structure-specific explainer, it is the difference between explaining every flow and explaining almost none. Disagreement is also a useful signal in itself, since it marks the flows closest to the decision boundary, where an analyst's attention is most valuable.

C. What a Surrogate Explanation Does and Does Not Show

A surrogate's decision path explains the surrogate's own decision. For the 98.73% of flows on which both models agree, it gives a correct account of a rule that leads to that class label, but it does not guarantee that XGBoost used the same features to reach the label [11], [12]. For the 1.27% of flows on which the models disagree, the surrogate's explanation is not faithful to the deployed detector at all. The overlap between the surrogate's upper splits and the top SHAP features (sttl, ct_state_ttl, dload and ct_dst_sport_ltm) suggests that the surrogate captures the main structure of the teacher's decisions, but agreement of predicted labels does not prove feature-level equivalence. Fidelity should therefore be reported with every surrogate explanation, and disagreement cases handled explicitly.

D. What the Explanations Revealed about the Detector

The most consequential result of the attribution analysis was not an explanation of an individual alert but the discovery that the detector leaned heavily on TTL features. The robustness experiments quantified the consequences: the official partition cost about five F1 points, TTL removal a further 1.8–2.5, and multi-class detection of the rarest attack families remained weak (F1 0.541–0.625). Reported scores on random splits of UNSW-NB15 should therefore be read as upper bounds, and detectors intended for deployment should be validated on the official partition, with and without features that may encode testbed artefacts, and on other datasets.

E. Relevance to WSN and IoT Deployment

The experiments use general network-flow data and were run on the workstation described in Section III-G, not on sensor nodes or gateways. Absolute latencies will differ on constrained devices, but the ratios between detection, surrogate explanation, TreeSHAP and model-agnostic attribution are likely to remain of the same order, because they reflect the number of model evaluations or tree traversals each method needs. Validation on WSN-specific data such as WSN-DS [37] and on-device timing are needed before conclusions are drawn for WSNs.

F. Limitations

- Only one benchmark (UNSW-NB15) is studied. The random split and the official partition are both evaluated, but external-dataset validation remains necessary.
- The explanation analysis and latency measurements use the random-split detector. Explanations of the official-partition and TTL-ablated detectors were not analyzed.
- Explainer latency was averaged over 100 test flows on a single general-purpose workstation, not on WSN or IoT hardware.
- Multi-class detection was evaluated for the detector only; explanation cost and surrogate fidelity were measured for the binary task.
- The tiered scheme is projected from measured rates and has not been evaluated end to end.
- Robustness of the detector and of its explanations to adversarial manipulation was not examined, although post hoc explainers such as LIME and SHAP are known to be manipulable [38].

VI. CONCLUSION

This paper measured the cost of explaining an ML intrusion detector on UNSW-NB15. XGBoost gave the best test F1-score among four classifiers (0.9680) at 0.0591 ms per flow, although it was statistically indistinguishable from a random forest. Five explanation routes for this detector spanned more than four orders of magnitude in cost: permutation SHAP and KernelSHAP needed 200.43 and 154.8 ms per flow, LIME 8.63 ms, TreeSHAP 0.742 ms and a depth-6 decision-tree surrogate 0.0088 ms, the last with fidelity of 0.9873 to the detector. Reducing the permutation-SHAP budget lowered its cost roughly in proportion but made its attributions less stable, so tuning cannot close this gap. The SHAP explanations exposed a strong reliance on TTL features, and robustness tests confirmed that both this reliance and the random split inflate the headline score: F1 fell to 0.9184 on the official partition and to 0.9000 without TTL features, and multi-class macro-F1 was 0.750. For tree ensembles, TreeSHAP is practical for per-flow explanation at moderate flow rates; at higher rates, or for detectors without a structure-specific explainer, a surrogate with reported fidelity, backed by offline SHAP for the flows on which it disagrees with the detector, offers per-flow transparency at negligible cost. Future work will evaluate the tiered scheme end to end, measure latency on gateway hardware and extend the study to WSN-specific datasets and to the robustness of explanations under adversarial manipulation.

AUTHOR CONTRIBUTION STATEMENT

Conceptualization: Dr.Ch.Sudha Rani
Literature Review and Methodology design: Dr.Ch.Sudha Rani, Prof.R.Rekha
Software: Dr.Ch.Sudha Rani, Prof.R.Rekha
Validation: Prof.R.Rekha
Formal Analysis: Prof.R.Rekha
Investigation: Prof.R.Rekha
Resources: Dr.Ch.Sudha Rani

Data Curation: Prof.R.Rekha
Writing original draft preparation: Dr.Ch.Sudha Rani
Writing review and Editing: Dr.Ch.Sudha Rani
Visualization: Prof.R.Rekha
Supervision: Prof.R.Rekha
Conflict of interest: The authors declare no conflicts of interest.
Data availability statement: Data supporting these findings are available within the article, at https://doi.org/10.26562/ijirae.2026.v1311.01, or upon request.

Supplementary materials: The supplementary materials are available at https://doi.org/10.26562/ijirae.2026.v1311.01
Publisher's note: AM Publications, India. IJIRAE (International Journal of Innovative Research in Advanced Engineering) Journals stays neutral with regard to jurisdictional claims in published maps and institutional affiliations. All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.
https://ampublications-india.com/

REFERENCES

1. I.Butun,S.D.Morgera, and R.Sankar, "A survey of intrusion detection systems in wireless sensor networks," IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 266–282, 2014, https://doi.org/10.1109/SURV.2013.050113.00191
2. T.Zhukabayeva, Z.Ahmad, A.Adamova, N.Karabayev, and A.Abdildayeva, "An edge-computing-based integrated framework for network traffic analysis and intrusion detection to enhance cyber–physical system security in industrial IoT," Sensors, vol. 25, no. 8, 2395, 2025, https://doi.org/10.3390/s25082395
3. S.S.Tippannavar, Vanditha M, and P.Nayak, "Smart intrusion detection system for CAN network implemented using LSTM strategy," International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 10, no. 3, pp. 98–105, 2023, https://doi.org/10.26562/ijirae.2023.v1003.08
4. N.Mohale and I. C. Obagbuwa, "A systematic review on the integration of explainable artificial intelligence in intrusion detection systems to enhancing transparency and interpretability in cybersecurity," Frontiers in Artificial Intelligence, vol. 8, 1526221, 2025, https://doi.org/10.3389/frai.2025.1526221
5. O.Arreche, T.Guntur, and M.Abdallah, "XAI-IDS: toward proposing an explainable artificial intelligence framework for enhancing network intrusion detection systems," Applied Sciences, vol. 14, no. 10, 4170, 2024, https://doi.org/10.3390/app14104170
6. O.Arreche, T.R.Guntur, J.W.Roberts, and M.Abdallah, "E-XAI: evaluating black-box explainable AI frameworks for network intrusion detection," IEEE Access, vol. 12, pp. 23954–23988, 2024, https://doi.org/10.1109/ACCESS.2024.3365140
7. S.M.Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, pp. 4765–4774, https://doi.org/10.48550/arXiv.1705.07874.
8. M.T.Ribeiro, S.Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 1135–1144, https://doi.org/10.1145/2939672.2939778
9. S.M.Lundberg, G.Erion, H.Chen, A.DeGrave, J.M.Prutkin, B.Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," Nature Machine Intelligence, vol. 2, no. 1, pp. 56–67, 2020, https://doi.org/10.1038/s42256-019-0138-9
10. M.W.Craven and J.W.Shavlik, "Extracting tree-structured representations of trained networks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 8, 1996, pp. 24–30.
11. R.Guidotti, A.Monreale, S.Ruggieri, F.Turini, F.Giannotti, and D.Pedreschi, "A survey of methods for explaining black box models," ACM Computing Surveys, vol. 51, no. 5, Art. 93, 2018, https://doi.org/10.1145/3236009
12. C.Molnar, Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book/
13. R.Sommer and V.Paxson, "Outside the closed world: on using machine learning for network intrusion detection," in Proc. IEEE Symp. Security and Privacy, 2010, pp. 305–316, https://doi.org/10.1109/SP.2010.25
14. D.Arp, E.Quiring, F.Pendlebury, A.Warnecke, F.Pierazzi, C.Wressnegger, L.Cavallaro, and K.Rieck, "Dos and don'ts of machine learning in computer security," in Proc. 31st USENIX Security Symp., 2022, pp. 3971–3988. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/arp
15. N.Moustafa and J.Slay,"UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in Proc. Military Communications and Information Systems Conf. (MilCIS), Canberra, Australia, 2015, pp. 1–6, https://doi.org/10.1109/MilCIS.2015.7348942
16. N.Moustafa and J.Slay, "The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," Information Security Journal: A Global Perspective, vol. 25, no. 1–3, pp. 18–31, 2016, https://doi.org/10.1080/19393555.2015.1125974
17. Y.K.Kasongo and Y.Sun,"Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," Journal of Big Data, vol. 7, Art. 105, 2020, https://doi.org/10.1186/s40537-020-00379-6

18. V.N.Gavali and S.Sangve,"Anomaly network intrusion detection: a review," International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 2, no. 4, pp. 90–95, 2015.
19. K.Karthikeyan and R.Mala,"Artificial neural content techniques for enhanced intrusion detection and prevention system," International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 3, no. 4, pp. 96–104, 2016.
20. A.V.Brahmane and A.Amune, "A survey of dynamic distributed network intrusion detection using online Adaboost-based parameterized methods," International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 1, no. 9, pp. 256–262, 2014.
21. A.Gupta, A.Kumar, A.K.Singh, U.S.Yadav, I.Ali, and S.Dubey, "Network security intrusion detection methods combining optimization algorithms and neural networks," International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 11, no. 12, pp. 974–979, 2024, https://doi.org/10.26562/ijirae.2024.v1112.14
22. S.Neupane, J.Ables, W.Anderson, S.Mittal, S.Rahimi, I.Banicescu, and M. Seale, "Explainable intrusion detection systems (X-IDS): a survey of current methods, challenges, and opportunities," IEEE Access, vol. 10, pp. 112392–112415, 2022, https://doi.org/10.1109/ACCESS.2022.3216617
23. A.Nadeem, D.Vos, C.Cao, L.Pajola, S.Dieck, R. Baumgartner, and S. Verwer, "SoK: explainable machine learning for computer security applications," in Proc. IEEE European Symp. Security and Privacy (EuroS&P), 2023, pp. 221–240, https://doi.org/10.1109/EuroSP57164.2023.00022
24. M.Wang, K.Zheng, Y.Yang, and X.Wang, "An explainable machine learning framework for intrusion detection systems," IEEE Access, vol. 8, pp. 73127–73141, 2020, https://doi.org/10.1109/ACCESS.2020.2988359
25. S.Patil, V.Varadarajan, S.M.Mazhar, A.Sahibzada, N. Ahmed, O. Sinha, S. Kumar, K. Shaw, and K. Kotecha, "Explainable artificial intelligence for intrusion detection system," Electronics, vol. 11, no. 19, Art. 3079, 2022, https://doi.org/10.3390/electronics11193079
26. A.Warnecke, D.Arp,C.Wressnegger, and K.Rieck, "Evaluating explanation methods for deep learning in security," in Proc. IEEE European Symp. Security and Privacy (EuroS&P), 2020, pp. 158–174, https://doi.org/10.1109/EuroSP48549.2020.00018
27. E.Štrumbelj and I.Kononenko, "Explaining prediction models and individual predictions with feature contributions," Knowledge and Information Systems, vol. 41, no. 3, pp. 647–665, 2014, https://doi.org/10.1007/s10115-013-0679-x
28. H.Chen, I.C.Covert, S.M.Lundberg, and S.-I. Lee, "Algorithms to estimate Shapley value feature attributions," Nature Machine Intelligence, vol. 5, no. 6, pp. 590–601, 2023, https://doi.org/10.1038/s42256-023-00657-x
29. R.Mitchell, E.Frank, and G.Holmes, "GPUTreeShap: massively parallel exact calculation of SHAP scores for tree ensembles," PeerJ Computer Science, vol. 8, Art. e880, 2022, https://doi.org/10.7717/peerj-cs.880
30. L.Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001, https://doi.org/10.1023/A:1010933404324
31. D.H.Wolpert, "Stacked generalization," Neural Networks, vol. 5, no. 2, pp. 241–259, 1992, https://doi.org/10.1016/S0893-6080(05)80023-1
32. T.Chen and C.Guestrin, "XGBoost: a scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785–794, https://doi.org/10.1145/2939672.2939785
33. F.Pedregosa, G.Varoquaux, A.Gramfort, V.Michel, B.Thirion, O.Grisel, et al., "Scikit-learn: machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html
34. L.Breiman, J.H.Friedman, R.A.Olshen, and C.J.Stone, Classification and Regression Trees. Belmont, CA, USA: Wadsworth, 1984 (reprinted Boca Raton, FL, USA: CRC Press, 2017), https://doi.org/10.1201/9781315139470
35. Q.McNemar,"Note on the sampling error of the difference between correlated proportions or percentages," Psychometrika, vol. 12, no. 2, pp. 153–157, 1947, https://doi.org/10.1007/BF02295996
36. T.G.Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," Neural Computation, vol. 10, no. 7, pp. 1895–1923, 1998, https://doi.org/10.1162/089976698300017197
37. I.Almomani, B. Al-Kasasbeh, and M. Al-Akhras, "WSN-DS: a dataset for intrusion detection systems in wireless sensor networks," Journal of Sensors, vol. 2016, Art. 4731953, 2016, https://doi.org/10.1155/2016/4731953
38. D.Slack, S.Hilgard, E.Jia, S.Singh, and H. Lakkaraju, "Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods," in Proc. AAAI/ACM Conf. AI, Ethics, and Society (AIES), 2020, pp. 180–186, https://doi.org/10.1145/3375627.3375830