



Class Imbalance Strategies, Calibration, and Explanation Stability in Tree-Based Intrusion Detection for IoT Networks

Abdulkerim Oguzhan Alkan^{1,*}

¹ Ministry of Treasury and Finance of Türkiye, Ankara, Türkiye

ARTICLE INFO

Article history:

Received 23 June 2026

Received in revised form 6 August 2026

Accepted 21 September 2026

Available online 1 October 2026

Keywords:

Intrusion detection; Class imbalance;
Gradient boosting; Probability calibration;
Explainable artificial intelligence

ABSTRACT

Network intrusion detection for the Internet of Things is based on traffic dominated by a few attack families, while security-relevant classes are rare. The credibility of published results depends on evaluation choices that are seldom examined together. Treatments for class imbalance are typically validated within a single learner family on a fixed training set, leaving the interaction between strategy and learner unclear and exposing reported gains to seed dependence. This paper compares five class imbalance strategies using two tree-based learners across the binary, category, and attack-type levels of the CICIoT2023 benchmark. The training pools are sampled from the full population using ten independent seeds, and the configurations are compared using matched protocols and nonparametric tests. In the controlled comparison, class weighting is the strongest single mechanism for gradient boosting, significantly outperforming every alternative. The balance between weighting and synthetic oversampling varies across learner families. Probability calibration separates the two model families by more than an order of magnitude in expected calibration error, while TreeSHAP rankings remain robust to resampling. The findings support a learner-dependent view of imbalance treatment and position calibration and explanation stability as evaluation axes in their own right, with implications for threshold-based alerting and the forensic use of feature attributions.

1. Introduction

The Internet of Things has moved from the fringes to a default feature of operational environments, and its security failures now have implications well beyond the compromised device. Surveys of the field [1] describe security, privacy and trust as perennial and largely unresolved requirements for deployments of the Internet of Things. The Mirai botnet showed the scale at which weakly protected embedded devices can be aggregated into denial-of-service infrastructure, and its retrospective analysis remains the reference account of how inexpensive infection techniques can overwhelm well-defended targets [2]. Flooding attacks of this kind still dominate the defensive problem, and the survey tradition on their mitigation shows just how much engineering effort the

* Corresponding author.

E-mail address: abdulkerimoguzhanalkan@gmail.com

<https://doi.org/10.65069/jessd21202623>

© The Author(s) 2025 | Creative Commons Attribution 4.0 International License

volume attack problem has soaked up [3]. Intrusion detection systems are one of the few controls that can observe traffic without prior knowledge of the specific exploit, and their machine learning branch has generated a large literature that spans techniques, data sets, and evaluation practices [4, 5]. The appeal of the machine learning approach is easy to state. Flow records are inexpensive to collect at the boundary of a network. The statistics they contain discriminate between broad categories of behavior, and labeling of attack traffic, once a capture is instrumented, scales better than manual curation of signatures. These same properties encourage abuse. A detector that is too well fitted to the benchmark may be worse than a simpler rule set, because its mistakes come with the appearance of measurement, not the appearance of judgment. That literature also comes with a familiar methodological warning. Sommer and Paxson [6] showed that the assumptions that make machine learning work in other domains do not carry over to network intrusion detection. Base rates are extreme, concept drift is continuous, and evaluation on closed data sets encourages overstatement. Their account introduces the present study.

The first choice in the data set. The benchmarks are known to be flawed and have been for 20 years. It has been shown that the KDD Cup 99 collection contains redundant records and artifacts that artificially inflate reported accuracy [7]. Its descendants solved part of the problem and brought their own biases along. Redundant records were removed from NSL-KDD but the aging protocol mix of the original capture was preserved. CIC-IDS2017 included background traffic with realistic content for contemporary attacks but did not reflect the embedded device population that now generates most of the new connections [8], instead providing only one week of a traditional network. UNSW-NB15 introduced synthetic modern attack behavior through a traffic generator with the trade-offs that the generated traffic always contains [9, 10]. The present environment has a CICIOT2023 data set created by the Canadian Institute for Cybersecurity. It contains traffic from 105 real devices under 33 attack types and seven categories. It exports fixed window flows described by 46 features and contains 46.7 million labeled flows [11]. The community has adopted it quickly, but that very structure that makes it attractive also defines the difficulty. The most common class is several thousand times more common than the rarest. Benign traffic is a small minority at the binary level. Volume attacks push out the rare classes whose detection is most significant in a security operations context. Quantified with the imbalance ratio

$$IR = N_{max}/N_{min} \quad (1)$$

where N_{max} and N_{min} denote the largest and smallest class counts, the data set reaches values beyond five thousand to one at the attack type level. Under this structure the headline metric of the field, accuracy, collapses as a measure of progress, and the meaningful evaluation level becomes the multiclass problem of identifying the attack type rather than merely its presence.

The second evaluation layer deals with treatments of class imbalance. Synthetic minority oversampling. The default remedy in applied work is the synthetic minority oversampling introduced by Chawla *et al.*, [12]. It creates interpolations between nearest neighbors of the minority class. Batista *et al.*, [13] combined oversampling with data cleaning, leading to SMOTE-ENN and SMOTE-Tomek where a neighborhood filter removes samples with labels that do not coincide with their neighbors. He *et al.*, [14] assigned the budget of synthetic generation based on the learning difficulty of each minority region. At the objective level, the focal loss of Lin *et al.*, [15] reshapes the training signal such that easy majority examples have less contribution to the gradient. Reviews of ensembles and imbalanced data suggest that the interaction between learner and treatment is systematic rather than incidental as boosting and bagging absorbing class information differently [16]. The commonality of this literature is largely based on the evaluation pattern. The practitioner inherits a recommendation with an unclear scope. A treatment is validated within a single learner family, on a

small number of seeds and at a single level of granularity. The seed question is not a cosmetic one. Oversampling adds randomness because of the choice of neighbor pairs and a protocol that fixes the training data but varies only the seed of the learner measures a fraction of the variance faced at deployment. The question of granularity is equally important. A rare class rescued by a binary-level resampling scheme may be lost at the attack-type level, where the operational question of which response playbook to invoke is posed. A further complication is given less attention than it merits. Flow features are binary protocol indicators and integer counters. Plain interpolation can be used to place synthetic examples at fractional values that have no physical meaning, motivating type-aware variants, although very few studies quantify the effect.

The third layer is the learner itself. Random forests [17] introduced by Breiman combine randomized trees by means of majority voting and are still hard to beat on tabular features with modest tuning. Friedman [18] formalized gradient boosting, and Chen and Guestrin [19] scaled it up in XGBoost, which fits trees to the residual errors of an additive model, and has become the strongest general-purpose family for structured data. Both learners are well-represented in intrusion detection research, and dedicated surveys document the extent to which random forests have been mapped onto this domain [20]. The software ecosystem matters for reproducibility, and the scikit-learn environment of Pedregosa *et al.*, [21] supplies the reference implementation of the classical learners used throughout this study. Deep architectures, from the long short-term memory network of Hochreiter and Schmidhuber [22] to the attention mechanism of Vaswani *et al.*, [23], holds the record results on sequence and spatial structure. Surveys of deep learning for intrusion detection show both their strength and their cost, since training and tuning at the scale of modern flow exports requires hardware that tree ensembles do not [24, 25]. This study concentrates on the tree-based family not because the deeper models are uninteresting, but because the imbalance, calibration, and explanation questions posed here are precisely the ones a deployed flow-based detector faces first, and tree ensembles are the class of model most often actually deployed. The point is empirical as much as practical. Reported results on this benchmark place tuned gradient boosting within a narrow margin of far more expensive architectures at the flow level, while the tree models train on commodity hardware, expose feature importances directly, and tolerate the mixed continuous and binary feature space of flow exports without preprocessing gymnastics.

Evaluation design is the fourth layer, and it binds the others. Macro averaged scores, the Matthews correlation coefficient [26], and the Cohen kappa [27] each correct a different distortion of accuracy under skewed class priors. The macro average treats the classes equally and therefore registers progress on the rare classes that the weighted average hides. The Matthews coefficient compresses the full confusion matrix into a single correlation that remains informative under extreme imbalance, and the kappa discounts the agreement that chance alone would produce. None of the three, however, says anything about the reliability of the probabilities that accompany the predictions. Precision-recall analysis carries information that receiver operating curves lose under imbalance, as Saito and Rehmsmeier [28] demonstrate in detail. Calibration adds a separate axis on which a model can be accurate in the classification sense and still unreliable at the probability level, a distinction documented for modern classifiers by Guo *et al.*, [29] and operationalized in the expected calibration error of Naeini *et al.*, [30]. Finally, explanations accompany deployed models. Shapley additive explanations [31] attribute predictions to features through a consistent axiomatic framework, the explainable artificial intelligence literature has organized the requirements such explanations must meet [32], and reference treatments catalog the strengths and limits of the methods in practice [33]. Whether the explanations of a resampled model remain comparable to those of the original is an empirical question that the literature has not settled for network intrusion data, and it matters directly for the forensic use of feature attributions in incident reporting.

Three gaps follow from this reading. First, the interaction between imbalance strategy and learner family has not been measured at the attack type level on a current IoT benchmark under a protocol that redraws the training data for every seed. Second, calibration is rarely reported as an axis separate from accuracy in this literature, although threshold-based alerting depends on it. Third, the stability of explanation rankings under resampling is assumed rather than tested. This study addresses the three gaps together. It trains random forest and XGBoost configurations on ten independently drawn training pools from the CICIoT2023 population under a common, file-level split protocol, evaluates five imbalance strategies on XGBoost and three on random forest at three granularity levels, and compares configurations through Friedman and Wilcoxon-Holm procedures [34–36]. It reports calibration and feature-level explanations as first-class outputs, including the agreement of three feature selection methods and the stability of TreeSHAP rankings across resampling conditions. Section 2 describes the data and the protocol. Section 3 reports the results. Section 4 discusses their meaning and limits. Section 5 concludes.

2. Materials and Methods

2.1. Dataset

This study uses the CICIoT2023 dataset released by the Canadian Institute for Cybersecurity in 2023. The traffic was generated in a dedicated testbed that hosts 105 real IoT devices. Attack traffic was produced only by compromised devices and directed toward designated target devices, which preserves a clean separation between attacker and victim roles. Packet captures were converted into fixed window flow records described by 46 numeric features. Each record carries one of 33 attack type labels or the benign traffic label. Attack types are organized under seven categories, namely DDoS, DoS, reconnaissance, web-based attacks, brute force, spoofing, and Mirai. Adding the benign class gives eight classes at the category level and 34 classes at the attack type level.

The complete dataset was downloaded and scanned record by record to verify its integrity. The independently computed total of 46,686,579 flows matches the published documentation exactly, which confirms that the acquired copy is complete and uncorrupted. The class distribution is summarized in Table 1 and visualized on a logarithmic scale in Figure 1. The largest class, DDoS-ICMP_Flood, contains 7,200,504 flows and accounts for 15.42% of the data, whereas the smallest class, Uploading_Attack, contains only 1,252 flows. The resulting imbalance ratio computed through Eq. (1) reaches 5751 to 1. Benign traffic represents 2.35% of all records. At the category level, DDoS and DoS together exceed 90% of the total volume, as shown in Figure 2.

Table 1
CICIoT2023 dataset summary

Statistic	Value
Total flows	46,686,579
Number of classes	34
Number of features	46
Largest class	DDoS-ICMP_Flood (7,200,504; 15.42%)
Smallest class	Uploading_Attack (1,252; 0.003%)
Imbalance ratio (max to min)	5751 to 1
Benign share	2.35%
DDoS share	72.79%
DoS share	17.33%

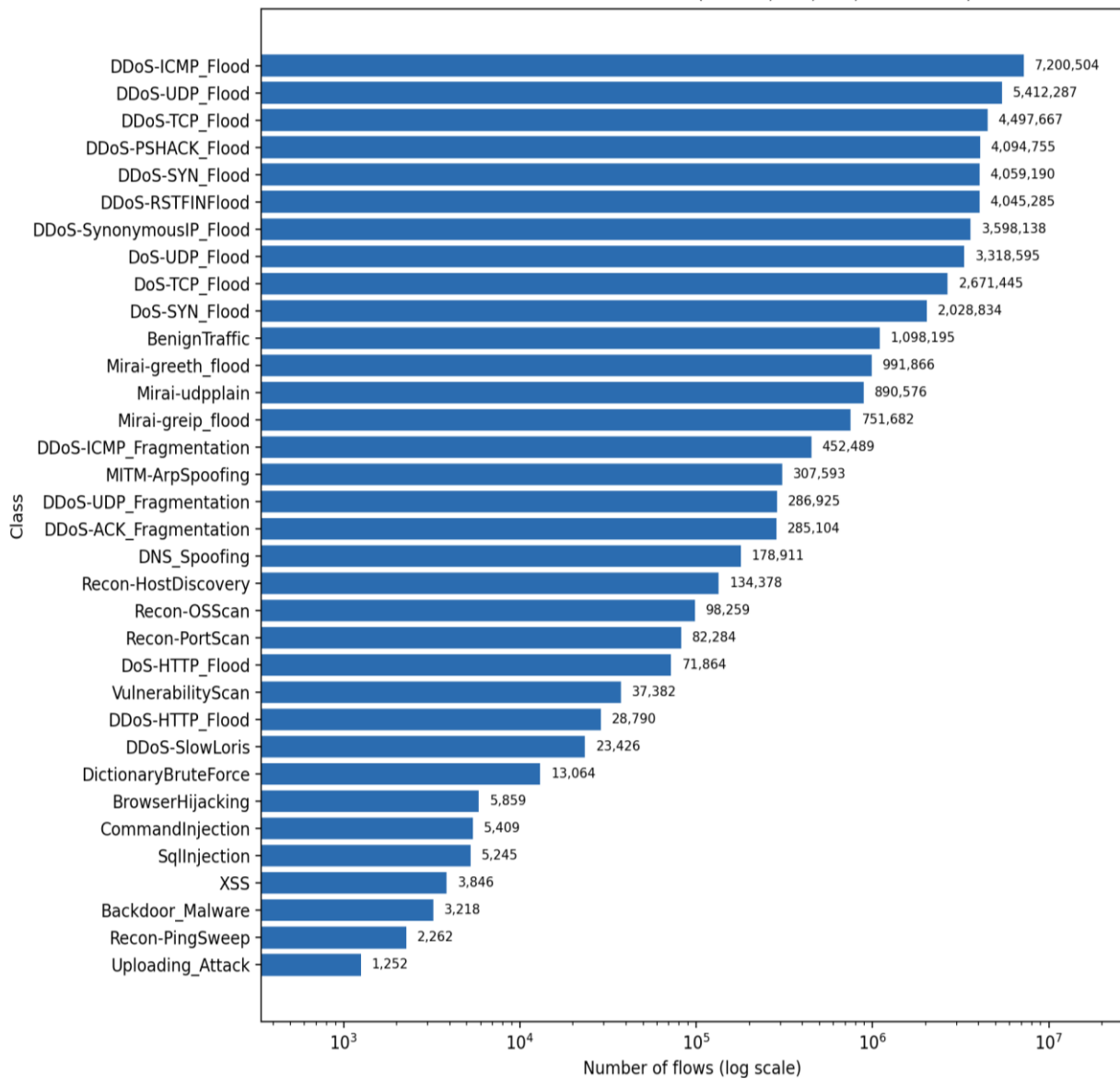


Fig. 1. CICIoT2023 class distribution (log scale)

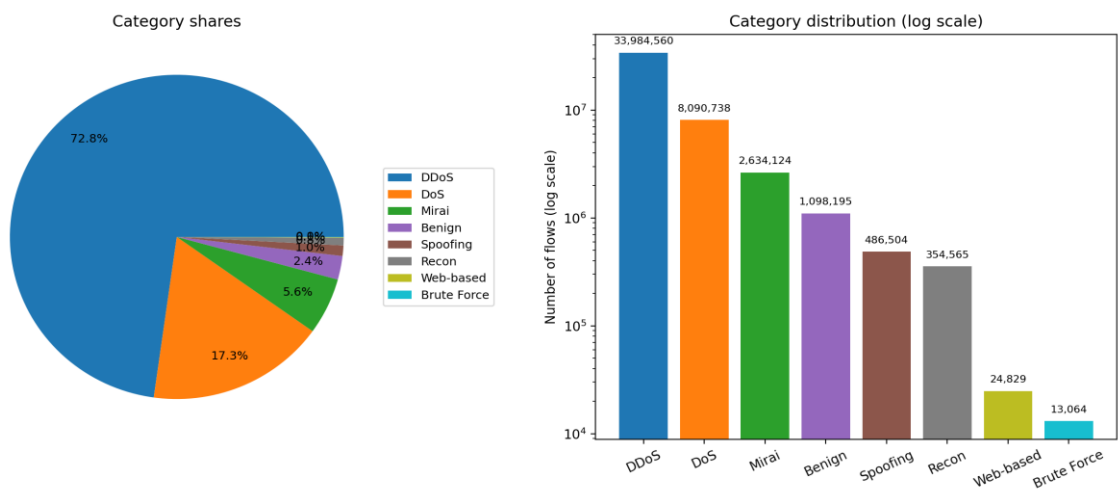


Fig. 2. CICIoT2023 category distribution

Three properties of this structure shape the design of the study. The benign class is a minority, which makes raw accuracy uninformative in the binary setting. Class level imbalance is extreme, which elevates per class recall in rare classes to a primary evaluation axis. The dominance of volume attacks inflates headline metrics unless macro level criteria are used. All evaluations in this work therefore report macro averaged scores, and the rarest classes receive a dedicated per class analysis with repetitions.

2.2. Preprocessing and Data Splitting

The producer removed incomplete packets during capture, and the dataset consequently contains no missing value. This property was verified on a random sample of 400,001 records drawn from three files. The sample contained zero missing values, zero malformed rows, and zero exact duplicate records. No imputation or duplicate removal step was applied.

Features were scaled with min-max normalization as

$$\tilde{x} = (x - \min) / (\max - \min) \quad (2)$$

where the minimum and maximum of each feature were estimated on the training subset alone and then applied unchanged to the test subset, so that no test information entered the scaling constants in Eq. (2).

To reduce direct file level mixing, the split was performed at the file level. Every fifth file of the 169 files was assigned to the test set and the remaining files to training. Because the released files do not contain scenario or session identifiers, this procedure cannot guarantee scenario level independence, and verbatim duplicates across partitions can remain. Section 4 quantifies this exposure as 2.22% of the test set.

Experiments run on capped stratified subsamples of the split files. For a class c with N_c available flows, the number of sampled records is $n_c = \min(N_c, C)$, where C is 15,000 for training and 4,000 for test. Within each class, records are drawn by reservoir sampling, which gives every flow of the class an equal inclusion probability. The cap bounds the compute budget of every downstream stage while it preserves the class ordering. It produces a fixed capped evaluation distribution rather than the natural prevalence, and all conclusions that depend on class priors are drawn with this in mind. Classes below the cap enter the sample in full.

The test sample is drawn once and held fixed across all repetitions, so that every configuration is evaluated on identical data. The training pool is redrawn from the population in each seed iteration with the seed of that iteration. The ten iterations therefore differ in their training data, which is what makes the seed repetitions independent measurements rather than reruns of an identical fit. Under this rule each training pool contains approximately 422,000 flows and the test sample contains 111,683 flows. The benign share rises from 2.35% of the population to about 3.5% in a training pool. This distortion is bounded, known in closed form, and it retains the attack dominant structure of the data. All resampling operations were applied exclusively to the training subset. The capped test distribution and its deviation from the population prevalence are documented class by class in Table 2, and metrics that depend on class priors are interpreted against this table.

Table 2
CICIoT2023 test-set class distribution and population shares

Class	Test flows	Test share	Population share
Mirai-udpplain	4,000	3.582%	1.908%
DDoS-PSHACK_Flood	4,000	3.582%	8.771%
DDoS-ICMP_Flood	4,000	3.582%	15.423%
DDoS-RSTFINFlood	4,000	3.582%	8.665%
Mirai-greip_flood	4,000	3.582%	1.610%

Table 2
Continued

Class	Test flows	Test share	Population share
DDoS-UDP_Flood	4,000	3.582%	11.593%
DoS-UDP_Flood	4,000	3.582%	7.108%
DDoS-TCP_Flood	4,000	3.582%	9.634%
DDoS-UDP_Fragmentation	4,000	3.582%	0.615%
DDoS-SYN_Flood	4,000	3.582%	8.695%
BenignTraffic	4,000	3.582%	2.352%
DoS-SYN_Flood	4,000	3.582%	4.346%
DDoS-ACK_Fragmentation	4,000	3.582%	0.611%
MITM-ArpSpoofing	4,000	3.582%	0.659%
DDoS-SynonymousIP_Flood	4,000	3.582%	7.707%
Mirai-greeth_flood	4,000	3.582%	2.125%
DoS-TCP_Flood	4,000	3.582%	5.722%
DoS-HTTP_Flood	4,000	3.582%	0.154%
DDoS-ICMP_Fragmentation	4,000	3.582%	0.969%
DNS_Spoofing	4,000	3.582%	0.383%
Recon-HostDiscovery	4,000	3.582%	0.288%
Recon-PortScan	4,000	3.582%	0.176%
VulnerabilityScan	4,000	3.582%	0.080%
Recon-OSScan	4,000	3.582%	0.210%
DDoS-SlowLoris	4,000	3.582%	0.050%
DDoS-HTTP_Flood	4,000	3.582%	0.062%
DictionaryBruteForce	2,478	2.219%	0.028%
BrowserHijacking	1,132	1.014%	0.013%
CommandInjection	1,073	0.961%	0.012%
SqlInjection	1,000	0.895%	0.011%
XSS	734	0.657%	0.008%
Backdoor_Malware	612	0.548%	0.007%
Recon-PingSweep	421	0.377%	0.005%
Uploading_Attack	233	0.209%	0.003%

2.3. Class Imbalance Strategies

Five strategies were defined and evaluated.

S0 is training without any intervention and serves as the reference.

S1 is SMOTE. For each minority sample, a nearest neighbor of the same class is selected at random among the k neighbors, and a synthetic sample is generated by linear interpolation through

$$x_{new} = x_i + \lambda \cdot (x_j - x_i), \lambda \sim U(0,1) \quad (3)$$

S2 is SMOTE-ENN. After SMOTE generation, an Edited Nearest Neighbour filter removes every sample, real or synthetic, whose class label disagrees with the majority of its neighbors. The filter targets noisy samples near class boundaries.

S3 is ADASYN. The synthetic generation budget is distributed in proportion to the learning difficulty of each minority sample, so that more synthetic samples are created near boundary regions.

S4 is class weighting. No resampling is performed. Random forest runs with balanced class weights, and XGBoost training uses balanced sample weights.

A controlled comparison of the five strategies follows a fixed protocol that is defined here in advance of any result. The base mirrors the population class distribution with a floor of twenty samples for the rarest classes, which keeps every class trainable while retaining the imbalance structure of the data. The base contains about 30,700 flows and is redrawn from the population with each of the ten seeds. All five strategies were evaluated on XGBoost. The main random forest

comparison included S0, S1, and S4. S2 and S3 were not evaluated with random forest because of computational constraints. Synthetic generation uses $k = 5$ neighbors for SMOTE and ADASYN, balancing to the majority count of the base, and the ENN stage uses three neighbors. Because the base is redrawn per seed, absolute values in this comparison are not comparable to Table 5, which uses pools of 422,000 flows. The purpose of the comparison is the ordering of strategies under identical conditions.

One property of the feature space constrains the synthetic methods. Protocol indicators and flag counters are binary or integer valued, and direct linear interpolation in Eq. (3) can assign fractional values to them. A type aware variant that rounds integer columns and clips binary indicators after generation was specified for future runs, and the experiments reported here applied plain interpolation. Synthetic records make up about 17% of the SMOTE training mass, and the results involving SMOTE were obtained under conventional feature space interpolation rather than a type aware variant.

2.4. Classification Models

Two families of tree-based classifiers were trained within each seed iteration on the same freshly drawn training pool of approximately 422,000 flows. A random forest with 100 trees was fitted with the Gini criterion, `max_features` equal to the square root of the number of features, `min_samples_leaf` of one, bootstrap sampling, and `max_samples` = 120,000, which gives every training flow a nonzero inclusion probability in each tree while peak memory remains bounded. XGBoost was fitted with 300 trees, depth 8, learning rate 0.1, the hist tree construction method, the softmax multiclass objective with probabilistic output, `subsample` = 1 and `colsample_bytree` = 1, and default L2 regularization. No early stopping was used, and no validation set existed in this protocol. Both learners therefore draw on the identical training pool within every iteration, and the comparison between them is balanced by construction. The full grid of six configurations was repeated with ten independent seeds, each with its own training pool drawn from the population.

2.5. Feature Selection and Explainability

The original capture contains a timestamp field. The released flow records used here contain no timestamp column, no address, and no port field, and the 46 numerical features were used as predictors. No explicit timestamp, address, or port feature was used. This excludes these direct identifiers, but it does not eliminate dependence caused by overlapping windows or shared captures. All importance and selection statistics were computed from training data only. Mutual information measures the divergence between each feature and the class label and was computed on a subsample of 150,000 flows drawn from the training pool of the corresponding iteration. The XGBoost gain score reports the total contribution of each feature to split decisions. TreeSHAP estimates Shapley values for every feature, and global importance was obtained by averaging absolute values over samples and classes through

$$\varphi_j = \sum_S |S|! (M - |S| - 1)! / M! \cdot [v(S \cup \{j\}) - v(S)] \quad (4)$$

The intersections of the three rankings were examined at cutoffs of 10, 15, and 20 features.

The stability of explanations under resampling was measured with the Spearman rank correlation between the TreeSHAP rankings of the S0 and S1 models. In addition, the five features with the highest mean absolute Shapley value were extracted for each class. These lists serve as flow level behavioral fingerprints of the attack families.

2.6. Evaluation Metrics and Statistical Analysis

The primary metric is the macro F1 score. Supporting metrics are macro precision, macro recall, Matthews correlation coefficient, Cohen kappa, ROC-AUC, and PR-AUC. Calibration was measured with the ten-bin expected calibration error of

$$ECE = \sum_b |B_b|/n \cdot |acc(B_b) - conf(B_b)| \quad (5)$$

Minority oriented evaluation tracks the mean per class recall over the eight rarest classes.

Multiclass ROC-AUC and PR-AUC use one versus rest averaging. The expected calibration error in Eq. (5) uses ten equal width bins over the maximum predicted probability, without prior correction.

Holm correction was applied separately to the prespecified XGB-S4 comparison family and to the nine pair cross family comparison matrix. Differences among configurations were examined with the Friedman test. The unit of the seed-based tests is the attack type level macro F1 of one independent training pool repetition, which gives ten blocks. The fixed test set means that these intervals reflect training pool and model randomness only, and 95% Student t confidence intervals across the ten pool estimates accompany the headline metrics. Pairwise comparisons against the XGB-S4 configuration used the Wilcoxon signed rank test with Holm correction. The three granularity levels are retained for reporting but are not used as test blocks, since they derive from the same model predictions.

2.7. Experimental Environment

All experiments ran on a dual core Intel Xeon Platinum processor. The software stack consisted of Python 3.12, scikit-learn 1.9.1, and XGBoost 3.4.1. The SMOTE, SMOTE-ENN, and ADASYN implementations were coded within this study and executed with a common subsampling core. The full dataset was used, with a compressed size of 2.98 GB and an extracted size of 13 GB. Seeds were fixed, and the sampling and training scripts will be released as supplementary material.

3. Results

3.1. Class Distribution and Imbalance Structure

Table 1 and Figure 1 show that the dataset carries imbalance at two distinct levels. At the binary level the benign share is 2.35%. At the attack type level, the gap becomes extreme. All web-based attacks together make up about 0.053% of the total, and brute force attacks sit at a similar scale. Each of these classes corresponds to a rare but high impact event category in a security operations context, which makes their recall practically relevant. The category view in Figure 2 makes the dominance of volume attacks explicit and justifies the macro-oriented evaluation protocol.

3.2. Overall Classification Performance

Tables 3, 4, and 5 report the results at the binary, category, and attack type levels. The patterns are consistent across all three levels.

Table 3

Binary classification results (mean $\pm$ SD over ten pools)

Config	Accuracy	Attack precision	Attack recall	Attack F1	Benign recall	Macro recall
RF-S0	0.9855 $\pm$ 0.0002	0.9876 $\pm$ 0.0002	0.9975 $\pm$ 0.0001	0.9925 $\pm$ 0.0001	0.6619 $\pm$ 0.0044 [0.6588, 0.6651]	0.8297 $\pm$ 0.0022 [0.8281, 0.8313]
RF-S1	0.9838 $\pm$ 0.0002	0.9857 $\pm$ 0.0002	0.9977 $\pm$ 0.0001	0.9917 $\pm$ 0.0001	0.6100 $\pm$ 0.0051 [0.6063, 0.6137]	0.8039 $\pm$ 0.0026 [0.8020, 0.8057]
RF-S4	0.9845 $\pm$ 0.0001	0.9865 $\pm$ 0.0001	0.9976 $\pm$ 0.0001	0.9920 $\pm$ 0.0001	0.6312 $\pm$ 0.0037 [0.6286, 0.6338]	0.8144 $\pm$ 0.0018 [0.8131, 0.8157]
XGB-S0	0.9894 $\pm$ 0.0001	0.9929 $\pm$ 0.0001	0.9962 $\pm$ 0.0001	0.9945 $\pm$ 0.0000	0.8079 $\pm$ 0.0032 [0.8056, 0.8102]	0.9020 $\pm$ 0.0016 [0.9009, 0.9032]

Table 3
Continued

Config	Accuracy	Attack precision	Attack recall	Attack F1	Benign recall	Macro recall
	0.9897 ± 0.0001	0.9930 ± 0.0001	0.9963 ± 0.0001	0.9946 ± 0.0001	0.8115 ± 0.0038	0.9039 ± 0.0019
XGB-S1	0.0001	0.0001	0.0001	0.0001	[0.8087, 0.8142]	[0.9025, 0.9052]
XGB-S4	0.9898 ± 0.0001	0.9931 ± 0.0001	0.9964 ± 0.0001	0.9947 ± 0.0001	0.8136 ± 0.0031	0.9050 ± 0.0015
	0.0001	0.0001	0.0001	0.0001	[0.8114, 0.8158]	[0.9039, 0.9061]
Config	MCC	Kappa	ROC-AUC	PR-AUC	Macro F1 (95% CI)	
RF-S0	0.7681 ± 0.0028	0.7580 ± 0.0030	0.9924 ± 0.0002	0.9997 ± 0.0000	0.9925 ± 0.0001	[0.9924, 0.9926]
RF-S1	0.7372 ± 0.0033	0.7219 ± 0.0037	0.9901 ± 0.0004	0.9995 ± 0.0000	0.9917 ± 0.0001	[0.9916, 0.9917]
RF-S4	0.7492 ± 0.0021	0.7364 ± 0.0023	0.9915 ± 0.0002	0.9996 ± 0.0000	0.9920 ± 0.0001	[0.9919, 0.9920]
XGB-S0	0.8410 ± 0.0014	0.8401 ± 0.0015	0.9965 ± 0.0001	0.9999 ± 0.0000	0.9945 ± 0.0000	[0.9945, 0.9946]
XGB-S1	0.8447 ± 0.0018	0.8437 ± 0.0019	0.9968 ± 0.0001	0.9999 ± 0.0000	0.9946 ± 0.0001	[0.9946, 0.9947]
XGB-S4	0.8473 ± 0.0018	0.8463 ± 0.0018	0.9968 ± 0.0000	0.9999 ± 0.0000	0.9947 ± 0.0001	[0.9947, 0.9948]

Note: Values are mean ± SD across ten pools; brackets give 95% Student-t CIs where shown. Attack precision, recall, and F1 refer to the attack class.

Table 4
Category-level classification results (8 classes; mean ± SD over ten pools)

Config	Accuracy	Macro precision	Macro recall	Macro F1	MCC
RF-S0	0.9511 ± 0.0006	0.9069 ± 0.0009	0.8714 ± 0.0018	0.8859 ± 0.0014	0.9351 ± 0.0007
RF-S1	0.9383 ± 0.0005	0.8878 ± 0.0009	0.8565 ± 0.0012	0.8589 ± 0.0012	0.9190 ± 0.0006
RF-S4	0.9469 ± 0.0005	0.9005 ± 0.0012	0.8592 ± 0.0014	0.8756 ± 0.0014	0.9296 ± 0.0006
XGB-S0	0.9601 ± 0.0003	0.9115 ± 0.0008	0.9009 ± 0.0008	0.9057 ± 0.0007	0.9470 ± 0.0004
XGB-S1	0.9592 ± 0.0004	0.9079 ± 0.0010	0.9057 ± 0.0009	0.9046 ± 0.0008	0.9460 ± 0.0006
XGB-S4	0.9594 ± 0.0002	0.9084 ± 0.0007	0.9078 ± 0.0005	0.9050 ± 0.0005	0.9464 ± 0.0003
Config	Kappa	ROC-AUC	PR-AUC	ECE	Macro F1 (95% CI)
RF-S0	0.9349 ± 0.0007	0.9952 ± 0.0001	0.9368 ± 0.0008	0.0487 ± 0.0006	0.8859 ± 0.0014 [0.8849, 0.8869]
RF-S1	0.9182 ± 0.0006	0.9937 ± 0.0001	0.9222 ± 0.0010	0.0379 ± 0.0004	0.8589 ± 0.0012 [0.8580, 0.8597]
RF-S4	0.9293 ± 0.0006	0.9946 ± 0.0001	0.9291 ± 0.0006	0.0507 ± 0.0005	0.8756 ± 0.0014 [0.8746, 0.8766]
XGB-S0	0.9470 ± 0.0004	0.9975 ± 0.0000	0.9613 ± 0.0001	0.0016 ± 0.0003	0.9057 ± 0.0007 [0.9051, 0.9062]
XGB-S1	0.9458 ± 0.0006	0.9975 ± 0.0000	0.9612 ± 0.0004	0.0034 ± 0.0003	0.9046 ± 0.0008 [0.9039, 0.9052]
XGB-S4	0.9462 ± 0.0003	0.9976 ± 0.0000	0.9628 ± 0.0002	0.0035 ± 0.0002	0.9050 ± 0.0005 [0.9047, 0.9054]

Note: Values are mean ± SD across ten pools; brackets give 95% Student-t CIs for macro F1.

Table 5

Attack-type classification results (34 classes; mean $\pm$ SD over ten pools)

Config	Accuracy	Macro precision	Macro recall	Macro F1	MCC
RF-S0	0.9284 $\pm$ 0.0006	0.8965 $\pm$ 0.0015	0.8511 $\pm$ 0.0017	0.8638 $\pm$ 0.0019	0.9259 $\pm$ 0.0006
RF-S1	0.9198 $\pm$ 0.0006	0.8458 $\pm$ 0.0009	0.8772 $\pm$ 0.0010	0.8551 $\pm$ 0.0009	0.9172 $\pm$ 0.0006
RF-S4	0.9249 $\pm$ 0.0004	0.8993 $\pm$ 0.0010	0.8615 $\pm$ 0.0009	0.8752 $\pm$ 0.0009	0.9223 $\pm$ 0.0005
XGB-S0	0.9364 $\pm$ 0.0004	0.8846 $\pm$ 0.0008	0.8695 $\pm$ 0.0007	0.8755 $\pm$ 0.0007	0.9341 $\pm$ 0.0004
XGB-S1	0.9374 $\pm$ 0.0007	0.8733 $\pm$ 0.0013	0.8901 $\pm$ 0.0012	0.8802 $\pm$ 0.0012	0.9352 $\pm$ 0.0007
XGB-S4	0.9382 $\pm$ 0.0003	0.8770 $\pm$ 0.0008	0.8954 $\pm$ 0.0005	0.8844 $\pm$ 0.0006	0.9361 $\pm$ 0.0003
Config	Kappa	ROC-AUC	PR-AUC	ECE	Macro F1 (95% CI)
RF-S0	0.9259 $\pm$ 0.0006	0.9943 $\pm$ 0.0002	0.8930 $\pm$ 0.0015	0.0673 $\pm$ 0.0005	0.8638 $\pm$ 0.0019 [0.8625, 0.8652]
RF-S1	0.9170 $\pm$ 0.0006	0.9946 $\pm$ 0.0001	0.8955 $\pm$ 0.0009	0.0696 $\pm$ 0.0006	0.8551 $\pm$ 0.0009 [0.8545, 0.8557]
RF-S4	0.9222 $\pm$ 0.0005	0.9944 $\pm$ 0.0002	0.9014 $\pm$ 0.0009	0.0718 $\pm$ 0.0004	0.8752 $\pm$ 0.0009 [0.8745, 0.8758]
XGB-S0	0.9341 $\pm$ 0.0004	0.9978 $\pm$ 0.0000	0.9156 $\pm$ 0.0004	0.0018 $\pm$ 0.0002	0.8755 $\pm$ 0.0007 [0.8749, 0.8760]
XGB-S1	0.9352 $\pm$ 0.0007	0.9978 $\pm$ 0.0000	0.9287 $\pm$ 0.0005	0.0039 $\pm$ 0.0006	0.8802 $\pm$ 0.0012 [0.8793, 0.8811]
XGB-S4	0.9360 $\pm$ 0.0003	0.9980 $\pm$ 0.0000	0.9330 $\pm$ 0.0003	0.0029 $\pm$ 0.0002	0.8844 $\pm$ 0.0006 [0.8839, 0.8849]

Note: Values are mean $\pm$ SD across ten pools; brackets give 95% Student-t CIs for macro F1.

XGBoost attains higher scores than random forest at every level under the specified hyperparameters and matched training pools, and the margin is statistically significant at the attack type level. Both learners were trained on the identical pool within each iteration, so the margin is not an artifact of a data advantage. Gradient boosting draws class boundaries through a sequence of strong splits, whereas majority voting in random forest smooths those boundaries. Figure 3 makes this difference visible. The diagonal bands of XGB-S0 are narrow and intense, while the S1 panel shows a clear brightening of the diagonal cells in minority rows.

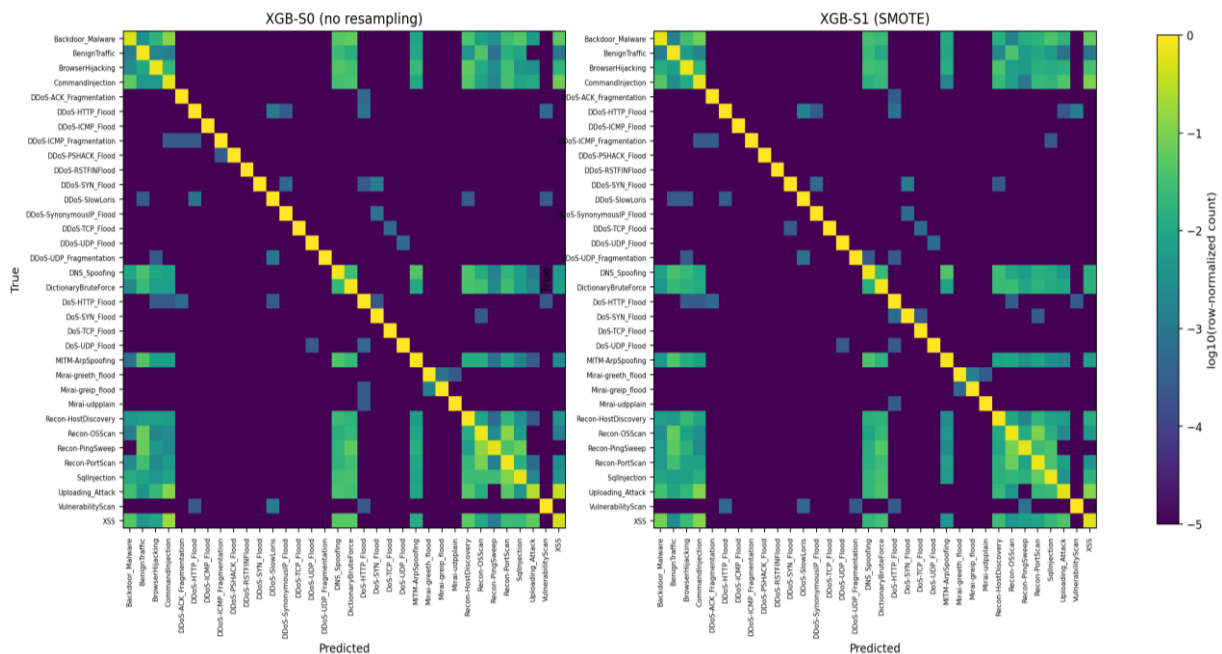


Fig. 3. Attack-type confusion matrices for XGB-S0 and XGB-S1

The gap between accuracy and macro F1 defines the reading of these tables. At the binary level the attack recall is above 0.996 for every configuration while the benign recall sits between 0.61 and 0.81, which means the detection rate is high but a substantial share of benign traffic is still flagged. The binary task is therefore not uniformly solved, and the operational burden sits in false alarms. At the attack type level macro F1 values remain in the high 0.80s for every configuration, so identifying the attack type is the harder problem. Security operations prioritize alert queues by attack type and route response teams accordingly. The attack type level is therefore the primary evaluation level of this study. The ROC curves in Figure 4 confirm the saturation of the binary task, with all four configurations nearly overlapping.

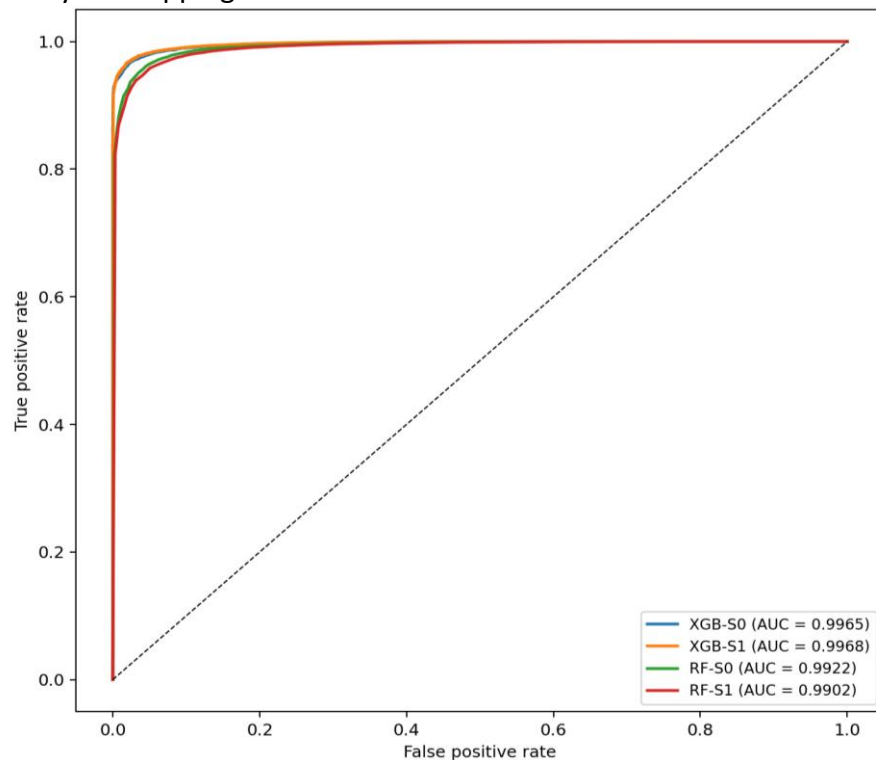


Fig. 4. Binary ROC curves for the principal configurations

3.3. Strategy Effects on Minority Classes

Table 6 reports per class recall in the eight rarest classes for a representative seed, and Figure 5 presents the same values as grouped bars. A single representative iteration is shown because per class breakdowns of this size are unreadable when averaged, and the macro level statistics in Tables 3 through 5 already establish the stability of the rankings across the ten seeds.

Table 6

Recall for the eight rarest classes (first seed)

Class	Test count	XGB-S0	XGB-S1	XGB-S4	RF-S0	RF-S1	RF-S4
Uploading_Attack	233	0.275	0.579	0.597	0.172	0.622	0.536
Recon-PingSweep	421	0.558	0.677	0.736	0.34	0.689	0.387
Backdoor_Malware	612	0.549	0.642	0.642	0.469	0.672	0.552
XSS	734	0.463	0.591	0.621	0.493	0.616	0.55
SqlInjection	1,000	0.686	0.764	0.792	0.597	0.729	0.583
CommandInjection	1,073	0.662	0.679	0.712	0.635	0.685	0.69
BrowserHijacking	1,132	0.736	0.76	0.774	0.644	0.723	0.587
DictionaryBruteForce	2,478	0.814	0.804	0.81	0.793	0.76	0.773

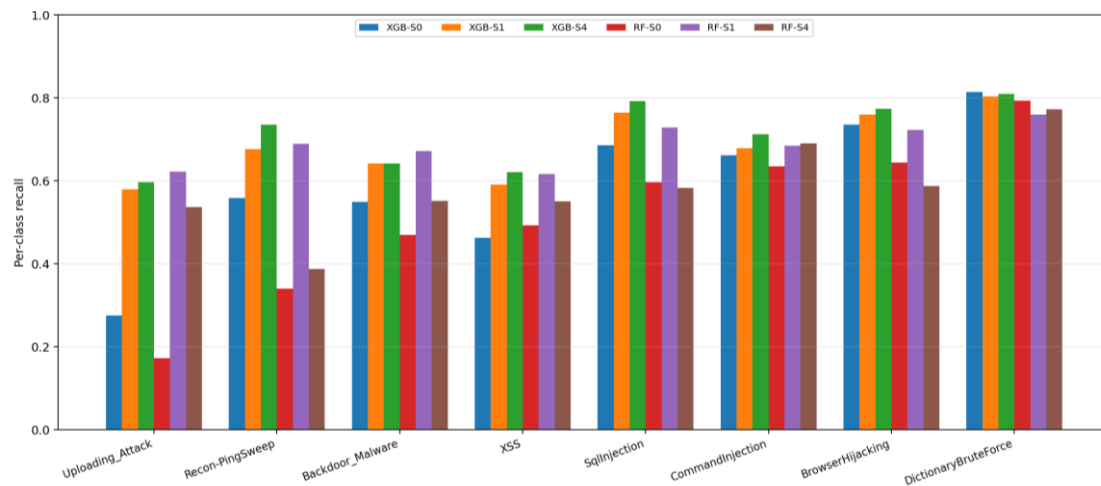


Fig. 5. Recall for the eight rarest classes by configuration

In the representative seed shown in Table 6, the gain in rare classes is purchased with a bounded cost in majority classes, and the direction of the trade depends on the model family. In XGBoost, class weighting leads SMOTE in seven of the eight classes with one tie, with SqlInjection and Recon-PingSweep as the clearest examples. In random forest, SMOTE surpasses weighting in most minority classes, including a strong recovery in Recon-PingSweep, while weighting shows an isolated regression in the same class. The overall random forest macro F1 under SMOTE still trails its weighted variant, which indicates that the synthetic density reaches rare classes at the expense of majority precision.

In this representative iteration the two families respond differently, which is consistent with their differently structured error surfaces. Gradient boosting reweights the full training set inside every tree, so class weighting enters the learning process naturally. Random forest trees are drawn from the original distribution and weighting only shifts node decisions. The additional density information supplied by synthetic samples is worth more in this second setting. The result supports the view that strategy effects depend on the learner family and caution against a one size fits all prescription.

3.4. Controlled Comparison of Resampling Strategies

Table 7 and Figure 6 compare the five strategies on a common training base. The base mirrors the population class distribution with a twenty sample floor for the rarest classes, contains about 30,700 flows, and is redrawn from the population in each of the ten seeds. The comparison therefore isolates the strategy effect under realistic imbalance.

Table 7

XGBoost strategy comparison on the controlled base (mean $\pm$ SD over ten pools)

Strategy	Macro F1	Macro recall	Macro precision	MCC	Mean recall, 8 rarest
S0	0.7053 $\pm$ 0.0040	0.7085 $\pm$ 0.0042	0.7310 $\pm$ 0.0051	0.8292 $\pm$ 0.0051	0.1361 $\pm$ 0.0076
S1	0.7943 $\pm$ 0.0049	0.7956 $\pm$ 0.0052	0.8093 $\pm$ 0.0057	0.8741 $\pm$ 0.0027	0.4138 $\pm$ 0.0210
S2	0.7939 $\pm$ 0.0040	0.7961 $\pm$ 0.0030	0.8072 $\pm$ 0.0045	0.8724 $\pm$ 0.0037	0.4231 $\pm$ 0.0119
S3	0.7939 $\pm$ 0.0036	0.7954 $\pm$ 0.0028	0.8091 $\pm$ 0.0038	0.8734 $\pm$ 0.0033	0.4154 $\pm$ 0.0127
S4	0.8135 $\pm$ 0.0070	0.8124 $\pm$ 0.0057	0.8336 $\pm$ 0.0083	0.8824 $\pm$ 0.0037	0.4689 $\pm$ 0.0256

The table exposes a decisive pattern, and it is statistically stable. The Friedman test on the ten pool repetitions rejects the equality of the five strategies with a chi square of 32.240 at $p = 1.71\text{e-}6$, and class weighting significantly outperforms every alternative after Holm correction at $p = 7.81\text{e-}3$. Without treatment the model collapses on the rare classes, with a mean recall of 0.136 across the

eight rarest classes. All four intervention strategies recover most of that loss, and they cluster within a narrow band of about 0.79 macro F1 and 0.41 to 0.47 mean rare class recall. Within the intervention cluster, class weighting leads on both axes simultaneously, ahead of SMOTE, SMOTE-ENN, and ADASYN, which do not separate clearly from each other. ADASYN slightly exceeds SMOTE only on the mean recall of the eight rarest classes. It does not improve the overall macro F1, macro recall, macro precision, or MCC. The variability is low for the overall metrics, whereas rare class recall shows greater seed to seed variability, with SD values up to 0.026.

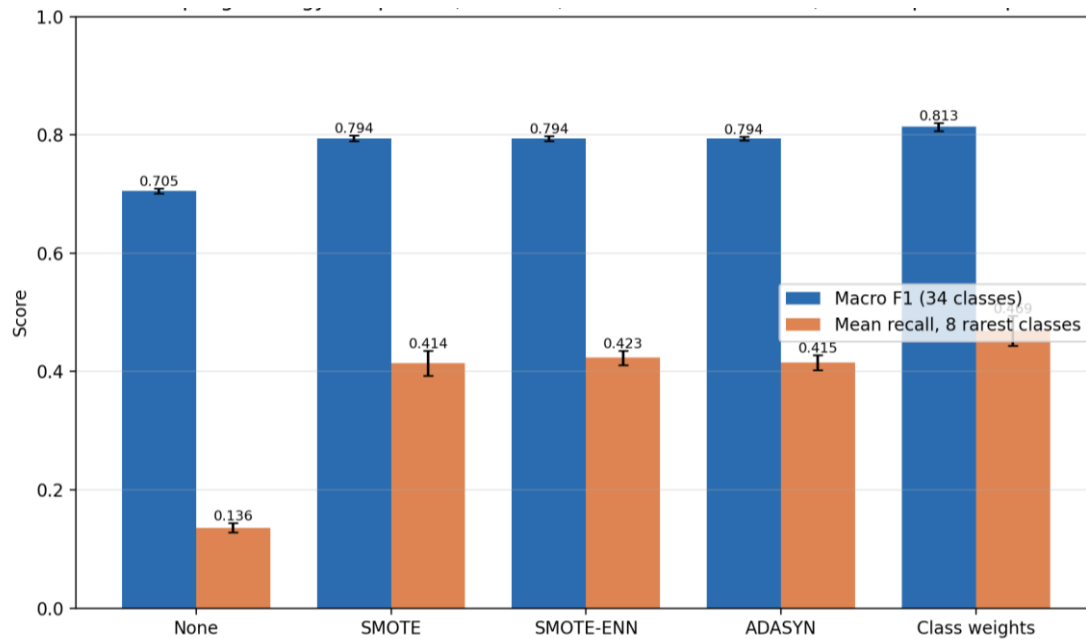


Fig. 6. Macro F1 versus recall for the eight rarest classes

The ordering refines a common assumption. Synthetic oversampling and hybrid cleaning carry a computational cost, yet on this data they do not purchase an advantage over simple weighting in the gradient boosting setting. The practical reading follows directly. Class weighting is the default choice for gradient boosting under this degree of imbalance, and the more expensive resampling machinery is justified when the target learner does not support instance weighting.

3.5. Calibration

ECE values appear in Tables 4 and 5. The calibration gap between the two model families is sharper than their performance gap. XGBoost configurations produce ECE values between 0.002 and 0.004 at the attack type level, while random forest configurations sit between 0.067 and 0.072. A gap of more than an order of magnitude of this size is operationally meaningful.

The source of the difference lies in how the outputs are produced. Random forest class probabilities are vote shares, and vote shares become overconfident away from the decision boundary. Gradient boosting works on a logistic output, and its probabilities decay more smoothly around the boundary. Threshold based alert rules behave very differently under the two output styles. In a well calibrated model the empirical frequency of alerts above a threshold matches the confidence level attached to those alerts. The quantitative footprint of this difference under natural prevalence evaluation is left to future work.

SMOTE has a limited effect on calibration. The ECE of XGB-S1 sits slightly above the S0 level. Synthetic samples appear to sharpen class probabilities. The gap remains small and does not change

deployment practice. In random forest, neither SMOTE nor weighting repairs calibration. Calibration emerges as a property that must be handled separately from class imbalance treatment.

3.6. Feature Selection Findings

Table 8 lists the top ten features under each selection method, and Figure 7 compares the normalized scores.

Table 8

Top-10 features by importance method

Rank	Mutual information	XGBoost gain	TreeSHAP
1	IAT	fin_flag_number	IAT
2	Max	ICMP	Number
3	Tot size	syn_flag_number	Protocol Type
4	Tot sum	UDP	Header_Length
5	Magnitue	psh_flag_number	syn_flag_number
6	AVG	Weight	flow_duration
7	Min	SSH	Min
8	Header_Length	HTTP	syn_count
9	Protocol Type	IAT	rst_count
10	rst_count	Number	Tot size

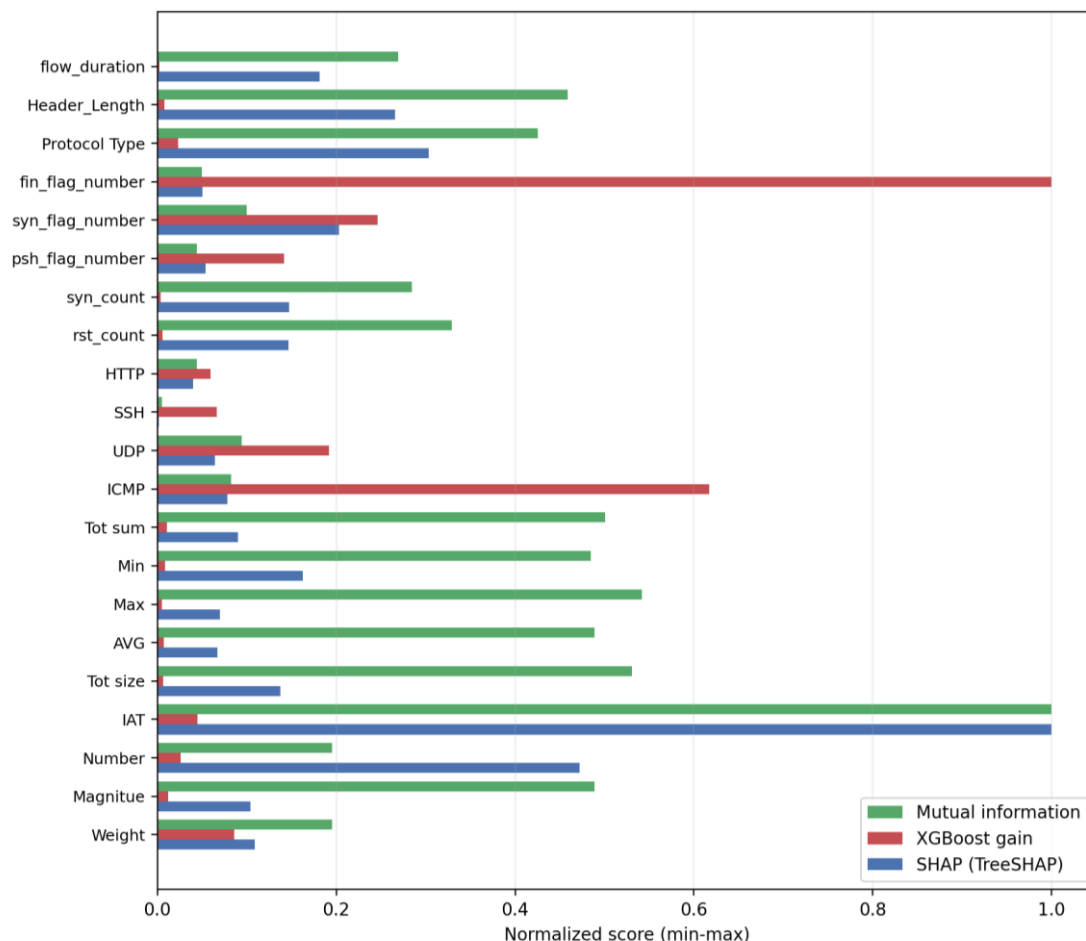


Fig. 7. Feature rankings by importance method

Only one feature appears in all three top ten lists. The count rises to four at the top fifteen cutoff and to eight at the top twenty cutoff. Agreement grows as the window widens. This pattern reflects

the different information sources of the methods. Mutual information measures marginal dependence, the gain score reflects usage in split decisions, and the Shapley value distributes prediction credit fairly. Strong alignment in the upper region is nonetheless present. IAT, Number, Protocol Type, Header_Length, and Min form the core of the selection.

The position of IAT at or near the top of all three rankings is the most informative finding of this section. Most attack families separate on timing structure. Volume attacks produce regularly spaced packet windows. Reconnaissance moves create sparse and irregular sampling. Spoofing attacks disrupt the client server rhythm. The result also carries an engineering implication. Derivative features that capture flow timing more richly are a promising direction for further gains.

3.7. Explainability and Forensic Interpretation

Figure 8 presents the global TreeSHAP ranking, and Figure 9 shows rank stability under resampling.

Rank stability reaches a Spearman coefficient of 0.994 and is statistically significant. Resampling preserves the explanation structure to a high degree. Movements in the top ranks are limited to a few positions, and larger shifts appear only in the mid tier. Finding matters for practice. Global TreeSHAP rankings were similar with and without SMOTE in this experiment, which supports reporting explanations of resampled models, although it does not by itself prove that every local explanation is preserved.

The class level lists in Table 9 are candidate behavioral indicators for forensic interpretation and require validation on independent traffic before operational use. The identification of Mirai-udpplain through the UDP protocol flag and timing features matches expected behavior. The prominence of SYN flag counters in DDoS-SYN_Flood agrees with the intuitive checks applied by network teams. The emergence of variance and flow duration features in SqlInjection is informative in a different way. Application layer attacks leave a weak packet level trace, and the model captures them from the statistical texture of the flow.

The fingerprint approach has its own boundary, visible in the same table. IAT and Number appear in most classes. These two features are strong discriminators but not class specific. On their own they do not constitute evidence in a forensic report. The class specific signature lies in the remaining three features. For DDoS-SYN_Flood the triplet of syn_flag_number, Max, and syn_count is the specific marker. For the spoofing classes the Header_Length and flow_duration pair must be read jointly.

Table 9

Class-level TreeSHAP fingerprints (top five features)

Class	Top five features
Mirai-udpplain	Number, IAT, UDP, Min, Protocol Type
MITM-ArpSpoofing	IAT, Header_Length, Number, syn_flag_number, flow_duration
DDoS-SYN_Flood	syn_flag_number, IAT, Number, syn_count, Tot sum
SqlInjection	Variance, Header_Length, flow_duration, IAT, syn_count
Recon-PortScan	Number, IAT, Weight, Header_Length, syn_count
DNS_Spoofing	Number, IAT, Header_Length, flow_duration, syn_flag_number
DictionaryBruteForce	IAT, Number, Header_Length, rst_count, syn_count
BenignTraffic	IAT, rst_count, Weight, Number, Header_Length

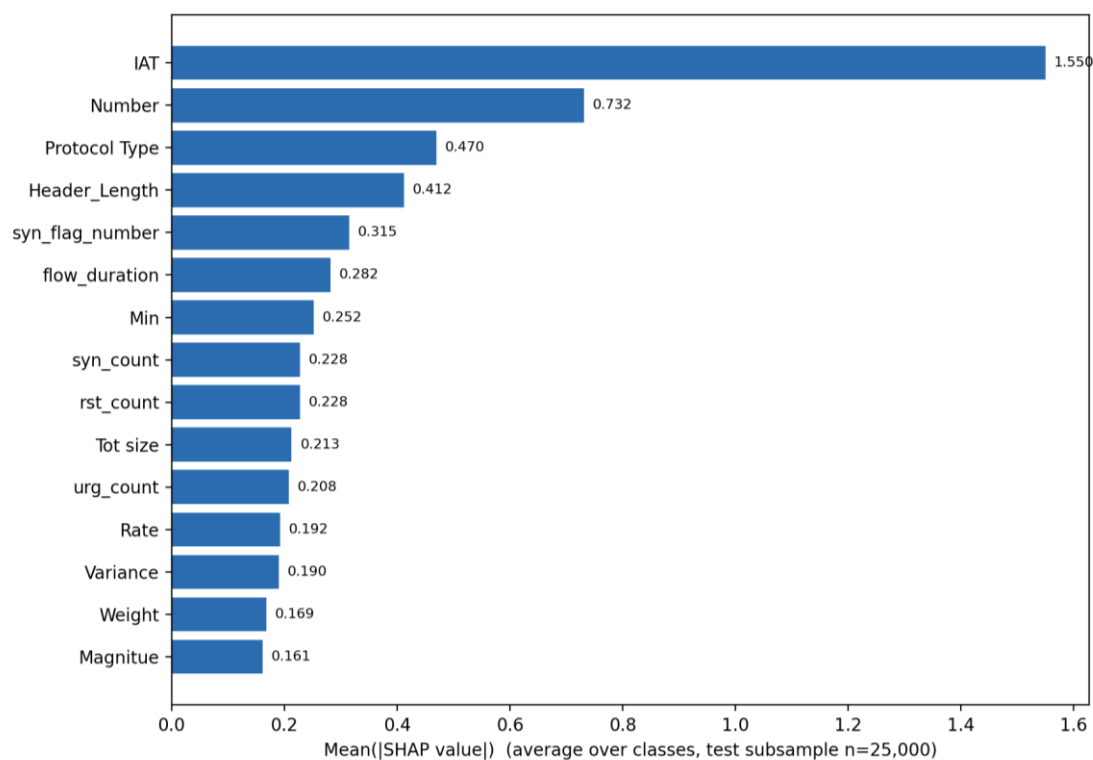


Fig. 8. Global TreeSHAP importance for XGB-S0

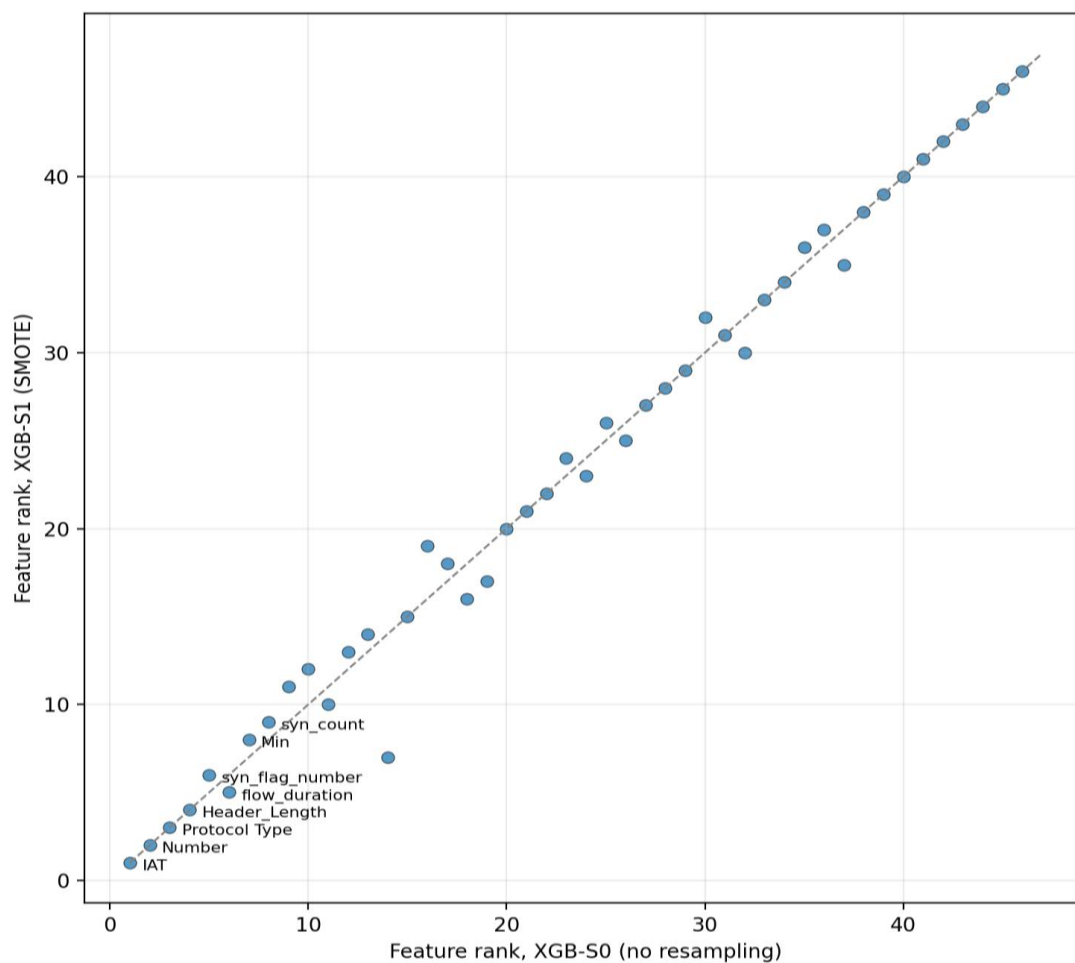


Fig. 9. TreeSHAP rank stability under resampling

3.8. Statistical Significance

Table 10 collects the statistical tests. The seed based tests use the ten independent training pool repetitions at the attack type level as blocks.

Table 10
Statistical tests of configuration and strategy differences

Test	Scope	Statistic	p_raw	p_Holm
Friedman	Six configurations; ten pool blocks at attack-type level	$\chi^2 = 48.571$	2.72×10^{-9}	
Wilcoxon-Holm, XGB-S4 vs XGB-S0	Ten paired blocks		1.95×10^{-3}	9.77×10^{-3}
Wilcoxon-Holm, XGB-S4 vs XGB-S1	Ten paired blocks		1.95×10^{-3}	9.77×10^{-3}
Wilcoxon-Holm, XGB-S4 vs RF-S0	Ten paired blocks		1.95×10^{-3}	9.77×10^{-3}
Wilcoxon-Holm, XGB-S4 vs RF-S1	Ten paired blocks		1.95×10^{-3}	9.77×10^{-3}
Wilcoxon-Holm, XGB-S4 vs RF-S4	Ten paired blocks		1.95×10^{-3}	9.77×10^{-3}
Friedman, extension	Five strategies; ten pool blocks	$\chi^2 = 32.240$	1.71×10^{-6}	
Wilcoxon-Holm, S4 vs S0/S1/S2/S3	Ten paired blocks		1.95×10^{-3}	7.81×10^{-3}
Wilcoxon-Holm, XGB-S0 vs RF-S0	Ten paired blocks		1.95×10^{-3}	1.76×10^{-2}
Wilcoxon-Holm, XGB-S0 vs RF-S1	Ten paired blocks		1.95×10^{-3}	1.76×10^{-2}
Wilcoxon-Holm, XGB-S0 vs RF-S4	Ten paired blocks		0.625	0.625
Wilcoxon-Holm, XGB-S1 vs RF-S0	Ten paired blocks		1.95×10^{-3}	1.76×10^{-2}
Wilcoxon-Holm, XGB-S1 vs RF-S1	Ten paired blocks		1.95×10^{-3}	1.76×10^{-2}
Wilcoxon-Holm, XGB-S1 vs RF-S4	Ten paired blocks		1.95×10^{-3}	1.76×10^{-2}
Wilcoxon-Holm, XGB-S4 vs RF-S0	Ten paired blocks		1.95×10^{-3}	1.76×10^{-2}
Wilcoxon-Holm, XGB-S4 vs RF-S1	Ten paired blocks		1.95×10^{-3}	1.76×10^{-2}

Note: p_raw is the unadjusted p-value and p_Holm is the Holm-adjusted p-value. The XGB-S4 vs RF-S4 pair appears in both correction families; its p_Holm is 9.77×10^{-3} in the prespecified family and 1.76×10^{-2} in the nine-pair cross-family family.

The Friedman test establishes strong differences among the six configurations on the primary evaluation level. The Wilcoxon procedure with Holm correction shows that class weighted XGBoost significantly outperforms every alternative configuration, including the unweighted and SMOTE variants of the same learner. Eight of the nine cross family comparisons remain significant after Holm correction, with the unweighted XGBoost against class weighted random forest comparison the sole exception. In the controlled strategy comparison, class weighting significantly outperforms each of the four alternative strategies on the macro F1 axis while also achieving the highest minority recall. The explanation stability result remains highly significant and rests on a far larger effective sample than the seed-based tests.

4. Discussion

All four substantive readings pass the most rigorous test in the protocol, consistent with the results. The first is about the level of the evaluation. The binary detection on this data set yields an attack recall above 0.996 in all configurations, and a benign recall between 0.61 and 0.81. Thus, a high detection rate is accompanied by a substantial false alarm burden. The models disagree on the multiclass attack type problem, which is the problem that a security operation actually faces when it routes alerts. Studies reporting only binary accuracy on this benchmark present a task of limited practical value and the multiclass level should be the primary evaluation level in future work on CICIOT2023.

The second reading is concerned with the learner family and the interaction of strategies. Under matched training pools, XGBoost configurations have higher point estimates than random forest

configurations at each level and eight out of nine cross-family comparisons are significant after Holm correction. The strongest single mechanism, class weighting within XGBoost, significantly outperforms each of the four alternatives in the controlled comparison. This refines the expectation that synthetic oversampling should dominate under extreme imbalance. This is not an unreasonable expectation. In the representative minority breakdown, random forests behave differently and recover classes that weighting misses in the synthetic route. This asymmetry is consistent with the way each learner consumes information from the class. Gradient boosting naturally incorporates an external weighting scheme into the optimization, as it re-weights the entire training mass at each iteration. Random forest trees, however, are fitted to bootstrap draws of the original distribution and benefit from the extra density provided by the synthetic examples. This finding is consistent with the ensemble literature on imbalanced data, where the value of a preprocessing step is dependent on the ensemble that consumes it [16], replacing a single prescriptive answer with a learner-dependent one.

The third reading is calibration. For the XGBoost configurations, the expected calibration error at the attack type level is between 0.002 and 0.004, while for the random forest configurations it is between 0.067 and 0.072. Such a gap is an order of magnitude larger and exceeds the performance gap in relative terms. It comes from the way of making the outputs. Vote share estimates are overconfident the farther away from the decision boundary, while the logistic outputs decay more smoothly around it. The threshold-based alerting is different for the two styles of output. The empirical frequency of alerts above a confidence threshold is equal to the stated confidence only in the calibrated case. Here we see the miscalibration of modern classifiers documented by Guo *et al.*, [29] in a non-neural setting. The remedy direction they discuss, post-hoc recalibration, is the natural companion to any deployment of the random forest configurations. Calibration should be a permanent part of intrusion detection evaluation, along with the discrimination metrics.

The fourth reading is on explanations. The TreeSHAP rankings of the weighted and unweighted XGBoost models are correlated at 0.994 and the top of the ranking is almost exactly preserved. The retention of the global attribution structure after the treatment supports the forensic use of feature attributions for incident documentation, allowing for explanations of resampled models to be reported with confidence. The feature selection comparison includes a qualification that guards against overreading of the result. Mutual information, split gain and Shapley attribution agree on a small core of timing and counting features and their agreement on the upper rank increases with the size of the retained window, which reflects the different quantities the three methods measure rather than inconsistency. The attribution lists at the class level are candidate behavioral indicators and must be validated on independent traffic prior to operational use.

These readings also compare the study to published pipelines on the same benchmark. These pipelines are based on binary accuracy, hybrid deep architecture and they score high on it. The present results point to the location of the remaining difficulty, and provide multiclass evidence hidden by binary figures. The difference in aim is as much methodological as it is architectural: A protocol that reports calibration and explanation stability alongside discrimination, that holds the test set fixed, and that redraws the training pool for every seed produces a smaller number that means more. The training pools, seeds and evaluation scripts are provided with this paper, so that the numbers can be regenerated exactly.

5.4. Limitations and Future work

Several limitations constrain the interpretation of these findings. The dataset is distributed as partition files without scenario or session identifiers. Although file-level splitting reduces direct file mixing, it cannot exclude shared capture windows across partitions. In an audit of one representative

training pool, exact feature-vector matches, rounded to four decimal places and identified without requiring matching labels, were found for 2.22% of the fixed test flows. The results should therefore be interpreted as in-distribution benchmark evidence, and independent scenario-level generalization is not claimed.

The capped evaluation distribution increases the benign share relative to the population prevalence. Accordingly, prior-sensitive metrics should be interpreted with reference to the documented test distribution rather than natural network prevalence. The hybrid strategies were evaluated with XGBoost only; therefore, the learner-dependent pattern observed for SMOTE and SMOTE-ENN in random forest is based on the minority-class analysis rather than a complete controlled strategy-by-learner comparison.

Per-class recall is reported for one representative iteration for readability, and per-class confidence intervals across pools were not estimated. Similarly, the TreeSHAP stability coefficient is a point estimate based on one pair of seed-42 models and a 25,000-flow subsample; uncertainty intervals for this coefficient remain unavailable. The synthetic-data experiments used conventional feature-space interpolation, which can produce fractional values in binary indicator columns. The reported SMOTE results should therefore be interpreted within this representation rather than as evidence for the superiority of type-aware oversampling. Finally, the evaluation covers a single dataset and does not establish transferability to independent benchmarks or live traffic with natural prevalence.

Future work should address these limitations in three complementary directions. First, capped pooling should be replaced or supplemented with streaming or full-population learning procedures that can exploit the complete dataset. Second, the controlled comparison should be extended to the full strategy-by-learner grid, including random-forest evaluations of the hybrid strategies and per-class uncertainty estimates across training pools. Type-aware oversampling methods, such as a carefully specified SMOTE-NC treatment, should also be evaluated before stronger conclusions are drawn about synthetic resampling.

Third, the complete protocol should be replicated on an independent DNS-focused benchmark and, where appropriate data access is available, on live network telemetry with natural class prevalence. Scenario- or session-grouped evaluation should be prioritized whenever the required identifiers are available. These experiments would clarify whether the observed performance, calibration, minority-recall, and explanation-stability patterns transfer beyond the present in-distribution benchmark.

5. Conclusions

This study evaluated class imbalance strategies for tree-based intrusion detection on the CICIOT2023 benchmark under a protocol that redraws the training pool for each of ten seeds and compares configurations through matched, nonparametric procedures. Class weighting significantly outperformed every alternative strategy for XGBoost and carried the best overall profile at the attack type level, while the advantage of synthetic oversampling over weighting in random forest confirms that strategy effects depend on the learner family. The two model families differ in calibration by more than an order of magnitude in expected calibration error, which establishes calibration as an evaluation axis separate from accuracy for threshold-based alerting. TreeSHAP rankings proved stable under resampling, which grounds the forensic use of feature attributions from resampled models. The contribution is a measured and reproducible evaluation layer for imbalance treatment in flow-based intrusion detection, not a new detection architecture.

Acknowledgement

This research was not funded by any grant.

Conflicts of Interest

The authors declare no conflicts of interest.

References

- [1] Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76, 146–164. <https://doi.org/10.1016/j.comnet.2014.11.008>
- [2] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., & Zhou, Y. (2017). Understanding the Mirai botnet. In 26th USENIX Security Symposium (USENIX Security 17) (pp. 1093–1110). <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>
- [3] Zargar, S. T., Joshi, J., & Tipper, D. (2013). A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Communications Surveys & Tutorials*, 15(4), 2046–2069. <https://doi.org/10.1109/SURV.2013.031413.00127>
- [4] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity*, 2(1), 20. <https://doi.org/10.1186/s42400-019-0038-7>
- [5] Buczak, A. L., & Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176. <https://doi.org/10.1109/COMST.2015.2494502>
- [6] Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE Symposium on Security and Privacy (pp. 305–316). IEEE. <https://doi.org/10.1109/SP.2010.25>
- [7] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (pp. 1–6). IEEE. <https://doi.org/10.1109/CISDA.2009.5356528>
- [8] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In ICISSP (Vol. 1, pp. 108–116). <https://doi.org/10.5220/0006639801080116>
- [9] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers & Security*, 86, 147–167. <https://doi.org/10.1016/j.cose.2019.06.005>
- [10] Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 Military Communications and Information Systems Conference (MilCIS) (pp. 1–6). IEEE. <https://doi.org/10.1109/MilCIS.2015.7348942>
- [11] Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIOT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, 23(13), 5941. <https://doi.org/10.3390/s23135941>
- [12] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>
- [13] Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1), 20–29. <https://doi.org/10.1145/1007730.1007735>
- [14] He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence) (pp. 1322–1328). IEEE. <https://doi.org/10.1109/IJCNN.2008.4633969>
- [15] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2980–2988). <https://doi.org/10.48550/arXiv.1708.02002>
- [16] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4), 463–484. <https://doi.org/10.1109/TSMCC.2011.2161285>
- [17] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- [18] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 1189–1232. <https://doi.org/10.1214/aos/1013203451>
- [19] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). <https://doi.org/10.1145/2939672.2939785>

- [20] Resende, P. A. A., & Drummond, A. C. (2018). A survey of random forest based methods for intrusion detection systems. *ACM Computing Surveys (CSUR)*, 51(3), 1–36. <https://doi.org/10.1145/3178582>
- [21] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. <https://jmlr.org/papers/v12/pedregosa11a.html>
- [22] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- [23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In 31st Conference on Neural Information Processing Systems (NIPS 2017). https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
- [24] Aldweesh, A., Derhab, A., & Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189, 105124. <https://doi.org/10.1016/j.knosys.2019.105124>
- [25] Thakkar, A., & Lohiya, R. (2021). A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. *Archives of Computational Methods in Engineering*, 28(4), 3211–3243. <https://doi.org/10.1007/s11831-020-09496-0>
- [26] Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2), 442–451. [https://doi.org/10.1016/0005-2795\(75\)90109-9](https://doi.org/10.1016/0005-2795(75)90109-9)
- [27] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37–46. <https://doi.org/10.1177/001316446002000104>
- [28] Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), e0118432. <https://doi.org/10.1371/journal.pone.0118432>
- [29] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International Conference on Machine Learning* (pp. 1321–1330). PMLR. <https://proceedings.mlr.press/v70/guo17a.html>
- [30] Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 29, No. 1). <https://doi.org/10.1609/aaai.v29i1.9602>
- [31] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* 30 (pp. 4765–4774). https://proceedings.neurips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html
- [32] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82–115. <https://doi.org/10.1016/j.inffus.2019.12.012>
- [33] Molnar, C. (2025). *Interpretable machine learning: A guide for making black box models explainable* (3rd ed.). <https://christophm.github.io/interpretable-ml-book/>
- [34] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 1–30. <https://jmlr.org/papers/v7/demsar06a.html>
- [35] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6), 80–83. <https://doi.org/10.2307/3001968>
- [36] Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2), 65–70. <https://doi.org/10.2307/4615733>