A Five-Run Comparison of Lightweight Machine Learning Models for IoT Intrusion Detection

Pradyumn Singh
John F. Kennedy High School
La Palma, California, US
contactprady@gmail.com

Russ Alizadeh
Cypress College
Cypress, California, US
rafaculty@gmail.com

Abstract

IoT intrusion-detection models are useful only if they detect attacks without becoming too costly to run. We compared Logistic Regression, Decision Tree, Random Forest, and XGBoost on binary CICIoT2023 intrusion detection using a cleaned file with 21,005,260 rows and 37 numeric features. The main experiment used five attack-resampling balanced outer runs: each outer run reused all 1,047,367 benign records and sampled an equal number of attack records, followed by five inner train-validation-test splits. XGBoost had the highest default-threshold F1-score in the main experiment (0.983109 ± 0.000205), followed by Decision Tree (0.982860 ± 0.000257), Random Forest (0.982237 ± 0.000371), and Logistic Regression (0.974215 ± 0.000272). Because the main outer runs share the benign records, we interpret the paired tests as evidence from repeated attack resampling rather than as five fully independent datasets. To address this limitation, we added a disjoint sensitivity analysis in which both benign and attack rows were resampled into non-overlapping 100,000-per-class outer subsets. The same general pattern remained: XGBoost reached 0.982572 ± 0.000612 F1, while Logistic Regression reached 0.974407 ± 0.000920. A feature-ablation check dropping `Number` and `HTTPS` caused only small F1 decreases and did not change the main ordering. Per-record inference latency on the main balanced test split ranged from 0.615 μs for Logistic Regression to 1.124 μs for Random Forest, with XGBoost at 0.647 μs. Overall, the results support tree-based models for this controlled binary benchmark, while also showing why latency, footprint, and dataset-specific shortcuts must be considered before making deployment claims.

Keywords

IoT Security, Cybersecurity, Intrusion Detection, Machine Learning, XGBoost

ACM Reference Format:
Pradyumn Singh and Russ Alizadeh. 2026. A Five-Run Comparison of Lightweight Machine Learning Models for IoT Intrusion Detection. In *Proceedings of International Journal of Secondary Computing and Applications Research (IJSCAR VOL. 3, ISSUE 3)*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.67149/yhjs2024.5/br0qqg04

IJSCAR VOL. 3, ISSUE 3

1 Introduction

IoT devices are now part of homes, factories, hospitals, cars, and public infrastructure. That spread makes them useful, but it also gives attackers more places to look for weak points. Many IoT devices are always online, lightly protected, and difficult to patch after deployment [3, 4, 7]. Intrusion detection is therefore an important layer of defense, especially when the hardware itself cannot easily be redesigned.

Machine-learning intrusion detection is appealing because it can learn traffic patterns that rule-based systems may miss [4, 8]. For IoT systems, however, accuracy is not the whole problem. A detector may need to run near the edge, where storage, memory, and latency are limited [3, 10]. This paper therefore treats deployment cost as an offline measurement target, not as a claim that the models have already been certified on edge hardware.

This study compares four common tabular models: Logistic Regression, Decision Tree, Random Forest, and XGBoost. We chose this group because it creates a practical range from a compact linear baseline to more flexible tree-based models. Logistic Regression is fast and easy to interpret, but it can only learn a linear decision boundary. The tree-based models can capture nonlinear interactions, which are often useful in cybersecurity datasets [6, 8, 10]. The question is not whether a flexible model can win once. The question is whether the gain is repeatable and large enough to justify the added cost.

CICIoT2023 is a strong benchmark for this comparison because it was designed around large-scale IoT security traffic, with 105 devices and 33 attacks across seven attack families [5, 9]. The main experiment in this paper uses a balanced binary version of the task, where all malicious traffic is grouped into one attack class. This makes the model-family comparison easier to interpret because the majority attack class does not dominate the result. We also include a supplementary imbalanced validation to check whether the main ranking still appears when the original attack-heavy distribution is sampled directly.

The research question is: do tree-based models consistently outperform Logistic Regression on binary IoT attack detection, and is the improvement worth the extra compute? The two hypotheses are straightforward. **H1**: tree-based models achieve higher F1-scores than Logistic Regression. **H2**: those gains come with higher deployment cost,

measured through latency, model size, and memory footprint.

The study contributes a repeated-run comparison of four lightweight tabular models on CICIoT2023, pairs predictive metrics with cost measurements, adds an imbalanced validation, and includes reviewer-driven sensitivity checks for shared benign rows and possible shortcut features. The goal is not to propose a new algorithm or a final deployment system. It is to make a careful binary benchmark comparison that is useful when deciding which model family deserves further testing in a practical IoT intrusion-detection pipeline.

2 Related Work

Recent reviews of IoT intrusion detection reach a similar conclusion: machine learning can improve detection, but the best model depends on the deployment setting [4, 7, 8]. Lightweight methods remain important because many IoT environments cannot support large deep-learning pipelines or expensive real-time processing [3, 10]. At the same time, benchmark studies often emphasize peak accuracy, which can hide whether a result is stable across repeated sampling and whether it remains practical under latency and footprint constraints.

CICIoT2023 has become a common dataset for IoT intrusion-detection research [9]. Prior work has reported high binary-detection performance on this dataset, so the purpose of this paper is not to claim a new state of the art. The original CICIoT2023 benchmark reported very high binary accuracy for Random Forest and deep neural networks [9]. Yiltirak and Ozturk compared several standard machine-learning algorithms and reported Random Forest as the strongest method, including 96.79% F1 for binary classification and lower F1 for the 7-class and 33-class settings [11]. Adewole et al. evaluated ensemble methods with rule induction and reported XGBoost as the strongest CICIoT2023 binary model in their setup, with 98.55% F1 [1]. Almahaqeri et al. reported a more optimized gradient-boosting pipeline with feature selection, where XGBoost reached 0.995 macro-F1 and 0.355 μs/sample inference latency for binary detection [2]. Table 1 summarizes these prior results alongside this paper's main and disjoint-sensitivity F1 scores for direct comparison.

This paper therefore takes a narrower approach than many prior studies. It compares four familiar tabular models under the same preprocessing, threshold tuning, repeated-run structure, and deployment-cost measurements. The main experiment uses attack-resampling outer runs, and the revision adds a disjoint sensitivity analysis where both benign and attack rows are resampled. This framing is more limited than a full multiclass benchmark, but it directly addresses whether the ranking among lightweight model families is stable in a controlled binary setting.

3 Experimental Setup

The study used the CICIoT2023 benchmark dataset [9]. The raw CSV file contained 45,019,243 rows. Duplicate removal discarded 24,013,983 rows, or 53.34% of the raw file, leaving 21,005,260 cleaned rows with 37 numeric features. The cleaned file contained 1,047,367 benign records and 19,957,893 attack records. Duplicate removal affected the attack labels unevenly: for example, 4,985,687 DDOS-ICMP-FLOOD rows, 3,216,938 DDOS-UDP-FLOOD rows, and 2,745,436 DDOS-TCP-FLOOD rows were removed, while only 4,006 benign rows were removed. Because the cleaned file was strongly attack-heavy, the main experiment used a balanced binary task to compare model families under controlled class proportions.

For each of five outer random seeds (101, 202, 303, 404, and 505), we built one balanced binary subset with all 1,047,367 benign rows and an equal number of sampled attack rows. Each outer run therefore contained 2,094,734 rows. This design should be described as *attack-resampling*, not as five fully independent datasets, because the benign records were reused across outer runs. Within each outer run, five inner stratified train-validation-test trials were created with different split seeds. The split proportions were 70% training, 15% validation, and 15% testing, giving 1,466,313 training rows, 314,210 validation rows, and 314,211 test rows per trial. This produced 25 trials per model and 100 total model fits across the four models.

To address the shared-benign limitation, we added a reviewer-requested disjoint sensitivity analysis. This analysis used five non-overlapping outer subsets with 100,000 benign rows and 100,000 attack rows per outer seed. It used three inner train-validation-test splits per outer seed, producing 15 trials per model. We also repeated this disjoint analysis after dropping `Number` and `HTTPS`, the two features that ranked highest in the Random Forest importance analysis, to test whether the main conclusion depended on these possible dataset-specific shortcuts.

Preprocessing was fitted inside each trial to avoid information leakage. Numeric features were converted to numeric values, infinite values were replaced with missing values, and missing values were imputed inside the training pipeline. Standardization was also fitted only on the training split and then applied to the validation and test splits. In other words, validation and test records did not affect the fitted preprocessing steps.

The comparison included Logistic Regression, Decision Tree, Random Forest, and XGBoost. Logistic Regression used the `lbfgs` solver with `C=1.0` and `max_iter=1000`. Decision Tree used the Gini criterion with `max_depth=12`. Random Forest used 100 trees, `max_depth=14`, `max_features=sqrt`, bootstrapping, and `n_jobs=-1`. XGBoost used 100 estimators, depth 6, learning rate 0.1, subsampling and column-sampling rates of 0.8, `logloss` evaluation, and `n_jobs=-1`. These were fixed modest configurations rather than a full hyperparameter search, so the results compare reproducible model configurations rather than each model family's maximum possible performance.

F1-score was the primary metric because it balances precision and recall. Accuracy, precision, recall, ROC-AUC,

Table 1: Selected CICIoT2023 results from prior work. Results are not directly comparable because preprocessing, feature selection, class balancing, tuning, and hardware differ across studies.

Study	Task	Reported result	Relevance to this paper
Pinto Neto et al. [9]	Binary and multiclass CICIoT2023 benchmark	Random Forest binary accuracy 99.68%; DNN binary accuracy 99.44%	Establishes that binary CICIoT2023 detection can reach near-ceiling accuracy.
Yiltirak and Ozturk [11]	33-class, 7-class, and binary CICIoT2023 classification	Random Forest binary F1 96.79%; 7-class F1 94.31%; 33-class F1 94.84%	Shows that performance depends strongly on task granularity.
Adewole et al. [1]	Binary ensemble IDS with rule induction	XGBoost CICIoT2023 F1 98.55%; accuracy 98.54%	Gives a close ensemble-method comparison point for binary detection.
Almahaqeri et al. [2]	Optimized binary, 8-class, and 34-class gradient-boosting IDS	XGBoost binary macro-F1 0.995; inference latency 0.355 μs/sample	Shows what is possible with optimized feature selection and tuning.
This paper	Controlled binary benchmark with repeated runs and cost reporting	XGBoost main F1 0.983109; disjoint sensitivity F1 0.982572	Focuses on repeated sampling, transparency, and model-family tradeoffs rather than peak optimization.

PR-AUC, false positives, false negatives, and threshold-optimized F1 were also recorded. For deployment cost, the notebook measured training time, inference time for the full held-out test split, per-record inference latency, serialized model size, and memory deltas during fitting and prediction. For the main attack-resampling experiment, paired tests were computed using the five outer-run means, but the resulting p-values are interpreted cautiously because the benign records are shared across outer runs. The disjoint sensitivity analysis provides a stronger check because both classes were resampled into non-overlapping outer subsets. Paired t-tests were reported with bootstrap confidence intervals. Wilcoxon signed-rank tests were also computed, but with only five paired outer observations their two-sided p-values are limited and should be read cautiously.

4 Results and Discussion

4.1 Main balanced experiment

Table 2 summarizes the cleaned dataset and the main evaluation setup. The full cleaned CICIoT2023 file was strongly attack-heavy, with 19,957,893 attack rows and 1,047,367 benign rows. Duplicate removal removed more than half of the raw rows, mostly from high-volume attack labels. We used balanced binary subsets for the main comparison so that the results would reflect model behavior rather than the original class ratio, while also reporting the shared-benign limitation directly.

The main performance results are shown in Table 3 and Figure 1. XGBoost had the highest default-threshold F1-score across the 25 balanced trials (0.983109 $\pm$ 0.000205). Decision Tree was very close (0.982860 $\pm$ 0.000257), followed by Random Forest (0.982237 $\pm$ 0.000371). Logistic Regression was lower at 0.974215 $\pm$ 0.000272. The gap was small in absolute terms, but it appeared consistently

Table 2: Dataset and evaluation setup for the main balanced experiment.

Item	Value
Raw CICIoT2023 rows	45,019,243
Rows removed as duplicates	24,013,983 (53.34%)
Cleaned CICIoT2023 rows	21,005,260
Features after cleaning	37
Full cleaned benign rows	1,047,367
Full cleaned attack rows	19,957,893
Main task	Balanced binary detection
Main outer-run design	Attack-resampling; benign rows reused
Rows per balanced outer run	2,094,734
Benign rows per balanced run	1,047,367
Attack rows per balanced run	1,047,367
Training rows per trial	1,466,313
Validation rows per trial	314,210
Test rows per trial	314,211
Outer runs / inner trials	5 / 5
Total repeated fits per model	25
Disjoint sensitivity check	100,000 benign + 100,000 attack rows per outer run

across the attack-resampling outer runs. Compared with Logistic Regression, the mean outer-run F1 improvements were 0.008894 for XGBoost, 0.008645 for Decision Tree, and 0.008022 for Random Forest.

These results support H1, but with an important design qualification. The paired t-tests used five outer-run means, but those outer runs are not fully independent because all of them contain the same benign records. They are therefore best interpreted as repeated attack-resampling comparisons. Under that framing, the pattern was still consistent: every tree-based model beat Logistic Regression in mean F1.

Table 3: Predictive performance across 25 balanced trials. Values are mean ± SD across repeated trials. p-values compare the five attack-resampling outer-run mean F1-scores for each tree-based model against Logistic Regression and should be interpreted cautiously because benign rows are shared across outer runs.

Model	Accuracy	Precision	Recall	F1-score	p vs. LR
Logistic Regression	0.974846	0.999277	0.950379	0.974215 ± 0.000272	–
Decision Tree	0.983119	0.998176	0.968007	0.982860 ± 0.000257	2.46×10^{-8}
Random Forest	0.982542	0.999662	0.965410	0.982237 ± 0.000371	3.32×10^{-8}
XGBoost	0.983370	0.998781	0.967922	0.983109 ± 0.000205	4.30×10^{-9}

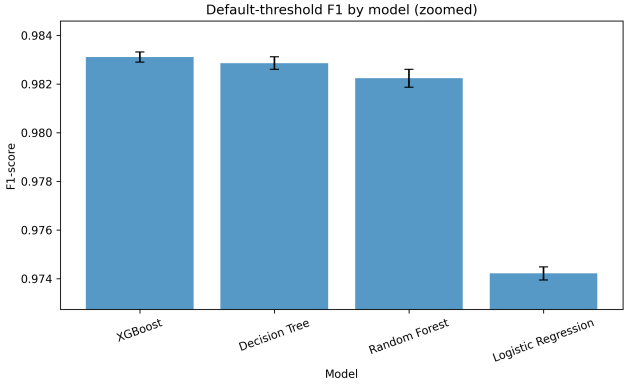

Figure 1: Default-threshold F1-score across 25 balanced trials, shown with a narrowed y-axis so the error bars are visible. Error bars show standard deviation across repeated trials.

Bootstrap 95% confidence intervals for the mean F1 improvement were fully above zero: 0.008515 to 0.008752 for Decision Tree, 0.007902 to 0.008134 for Random Forest, and 0.008817 to 0.008979 for XGBoost. The Wilcoxon signed-rank tests had the same direction, although the two-sided p-values were 0.0625 because there were only five paired outer-run observations. With $n = 5$ paired observations, 0.0625 is the smallest attainable exact two-sided Wilcoxon p-value, so this value reflects the resolution limit of the test rather than a change in the direction of the result. We therefore interpret the statistics as evidence of a stable advantage in this benchmark design, not as proof that the exact gap would hold in every IoT dataset.

4.2 Reviewer sensitivity checks

The disjoint sensitivity analysis addressed the main independence concern by resampling both classes. Each of five outer seeds used a non-overlapping subset with 100,000 benign and 100,000 attack rows, followed by three inner train-validation-test splits. Table 4 shows that the ranking remained similar to the main experiment. XGBoost again had the highest default F1-score (0.982572 ± 0.000612), while Logistic Regression remained lower (0.974407 ± 0.000920). The mean outer-run F1 improvements over Logistic Regression were 0.008165 for XGBoost, 0.007374 for Decision Tree, and 0.007093 for Random Forest. Bootstrap confidence intervals remained above zero: 0.007926 to 0.008428 for XGBoost, 0.007189 to 0.007522 for Decision Tree, and 0.006754 to 0.007432 for Random Forest. The disjoint result does not replace a full external validation dataset, but it reduces the concern that the original result came only from reusing benign records.

We also checked whether the high feature-importance rankings for `Number` and `HTTPS` were driving the result. Dropping those two features caused only small F1 decreases: -0.000171 for Logistic Regression, -0.000397 for Decision Tree, -0.001110 for Random Forest, and -0.000344 for XGBoost. The tree-based models still outperformed Logistic Regression after the ablation, with XGBoost maintaining the highest F1-score (0.982228 ± 0.000724). This does not prove that the models generalize to every dataset, but it suggests that the main conclusion is not dependent only on those two features.

Variance diagnostics also clarified the role of the outer-run structure. All variance values here refer to default-threshold F1-score. In the original attack-resampling experiment, the outer-seed variance of the outer-run means was smaller than the mean inner-split variance for every model: Logistic Regression 1.71×10^{-8} vs. 7.15×10^{-8}, Decision Tree 2.15×10^{-8} vs. 5.76×10^{-8}, Random Forest 2.08×10^{-8} vs. 1.44×10^{-7}, and XGBoost 9.19×10^{-9} vs. 4.13×10^{-8}. This supports the concern that reusing benign rows limited between-run variation. In the disjoint sensitivity analysis, the corresponding outer-vs.-inner variances were closer: Logistic Regression 5.23×10^{-7} vs. 5.57×10^{-7}, Decision Tree 4.03×10^{-7} vs. 4.07×10^{-7}, Random Forest 3.29×10^{-7} vs. 2.65×10^{-7}, and XGBoost 2.54×10^{-7} vs. 2.20×10^{-7}. This makes the disjoint analysis a better check on between-sample stability than the original attack-resampling design.

4.3 Supplementary imbalanced validation

The supplementary imbalanced validation tested whether the main conclusion depended too much on balancing. Each outer seed used a stratified 100,000-row sample with 95,014 attack records and 4,986 benign records, followed by three inner train-validation-test trials. Table 6 shows that the ranking changed somewhat, but the main message stayed similar. XGBoost again had the highest default-threshold F1-score (0.990388 ± 0.000448), with Random Forest close behind (0.990141 ± 0.000602). Decision Tree stayed above Logistic

Table 4: Reviewer-requested disjoint balanced sensitivity analysis. Each outer run used non-overlapping 100,000-row benign and 100,000-row attack subsets, with three inner splits per outer run. Values are mean ± SD across 15 trials per model.

Model	Accuracy	Precision	Recall	F1-score	Latency (μs/record)
Logistic Regression	0.975029	0.999262	0.950760	0.974407 ± 0.000920	0.438
Decision Tree	0.982053	0.996863	0.967151	0.981781 ± 0.000798	0.388
Random Forest	0.981824	0.999314	0.964311	0.981500 ± 0.000687	1.630
XGBoost	0.982842	0.998262	0.967369	0.982572 ± 0.000612	0.436

Table 5: Feature-ablation check on the disjoint sensitivity design after dropping `Number` and `HTTPS`.

Model	Full-feature F1	Ablated F1	ΔF1
Logistic Regression	0.974407	0.974236	-0.000171
Decision Tree	0.981781	0.981384	-0.000397
Random Forest	0.981500	0.980390	-0.001110
XGBoost	0.982572	0.982228	-0.000344

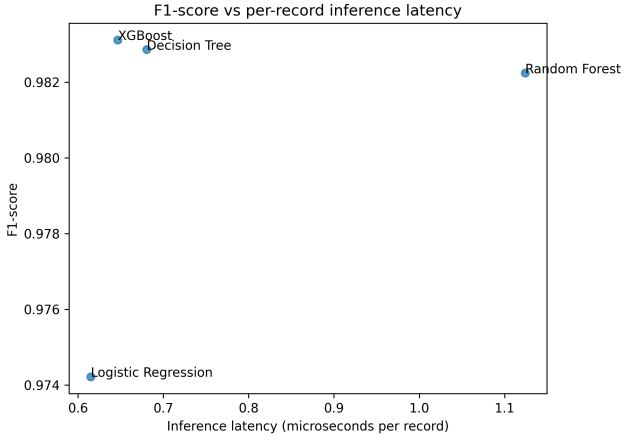

Figure 2: Performance-versus-latency tradeoff using per-record inference latency. The x-axis is log-scaled so the lower-latency models remain legible.

Regression, but the two ensemble methods separated more clearly from it than they did in the balanced experiment.

The imbalanced result is useful because it checks a different failure mode. In the balanced experiment, Decision Tree, Random Forest, and XGBoost were tightly grouped. In the imbalanced validation, XGBoost and Random Forest were more clearly ahead, and Logistic Regression again had lower recall. Compared with Logistic Regression, the mean outer-run F1 improvements were 0.016213 for XGBoost, 0.015966 for Random Forest, and 0.010198 for Decision Tree. The paired t-tests and bootstrap intervals again supported positive improvements over Logistic Regression. This does not replace testing on raw streaming traffic, but it does reduce the concern that the balanced results were only caused by artificial class proportions.

4.4 Deployment cost and operational behavior

The predictive ranking did not settle the cost question by itself. Table 7 and Figure 2 report both full-test-split inference time and per-record latency. The full balanced test split contained 314,211 rows, so the per-record latency is the clearest way to compare models. Logistic Regression was smallest and fastest per record, XGBoost was close behind with higher F1, and Random Forest was much larger and slower.

H2 was only partly supported. Tree-based models improved F1-score over Logistic Regression, but their costs differed sharply. Figure 4 shows the deployment footprint across models. Random Forest had the clearest cost penalty: it averaged 14.958 MB on disk and 1.124 μs per record on the balanced test split. Decision Tree was compact, but it did not beat XGBoost on F1. XGBoost was much smaller than Random Forest and had per-record latency close to Logistic Regression in this offline notebook benchmark. Because the timing was not measured on an edge device, this should be read as deployment-motivated evidence rather than a completed deployment benchmark.

The error counts, plotted in Figure 3, show why a small F1 gap can matter in security settings. Logistic Regression had the fewest false positives on average (108.00), but it missed the most attacks, with 7795.68 false negatives on the balanced test splits. Expressed per million attack flows, its false-negative rate corresponds to about 49,621 missed attacks. XGBoost missed about 32,078 attacks per million attack flows, Decision Tree about 31,993, and Random Forest about 34,590. In this benchmark, XGBoost therefore avoided roughly 17,543 missed attacks per million attack flows compared with Logistic Regression, although the exact operational impact would depend on the traffic volume and class mix in a real deployment.

Threshold tuning changed the ranking slightly. The optimized thresholds were 0.467 for Logistic Regression, 0.533 for Decision Tree, 0.316 for Random Forest, and 0.414 for XGBoost. At those thresholds, Random Forest had the highest optimized F1-score (0.983313), with XGBoost extremely close (0.983184). Because Random Forest was much heavier, the small tuned advantage would need to be justified by a

Table 6: Supplementary imbalanced validation on stratified 100,000-row samples. Values are mean ± SD across 15 trials per model. The samples preserve the original cleaned-data class distribution.

Model	Accuracy	Precision	Recall	Default F1	Optimized F1
Logistic Regression	0.952160	0.999946	0.949701	0.974175 ± 0.001032	0.985031
Decision Tree	0.970724	0.998657	0.970493	0.984373 ± 0.000975	0.988555
Random Forest	0.981280	0.990931	0.989354	0.990141 ± 0.000602	0.990105
XGBoost	0.981764	0.991985	0.988797	0.990388 ± 0.000448	0.990425

Table 7: Operational summary across 25 balanced trials. Batch inference time is measured on the full 314,211-row test split. Per-record latency normalizes that value to microseconds per record. FP and FN are false positives and false negatives averaged across test splits; Opt. F1 is threshold-optimized F1 using the validation-selected threshold.

Model	Train time (s)	Batch inf. (s)	μs/record	Size (MB)	FP	FN	Opt. F1
Logistic Regression	8.775 ± 0.510	0.193237	0.615	0.005	108.00	7795.68	0.974205
Decision Tree	12.471 ± 0.242	0.213913	0.681	0.066	278.00	5026.32	0.982852
Random Forest	27.693 ± 0.596	0.353203	1.124	14.958	51.24	5434.36	0.983313
XGBoost	7.410 ± 0.269	0.203198	0.647	0.263	185.64	5039.64	0.983184

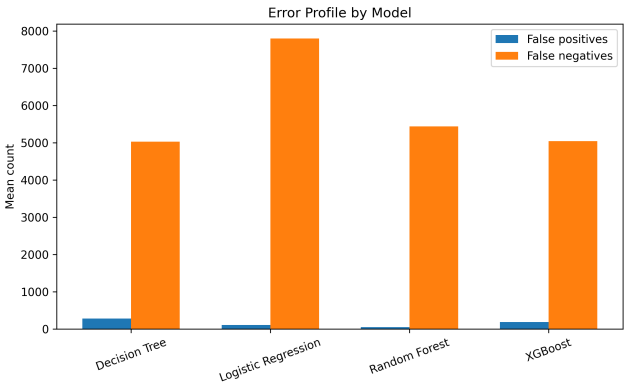

Figure 3: Error profile by model. Bars show mean false-positive and false-negative counts across the 25 balanced trials.

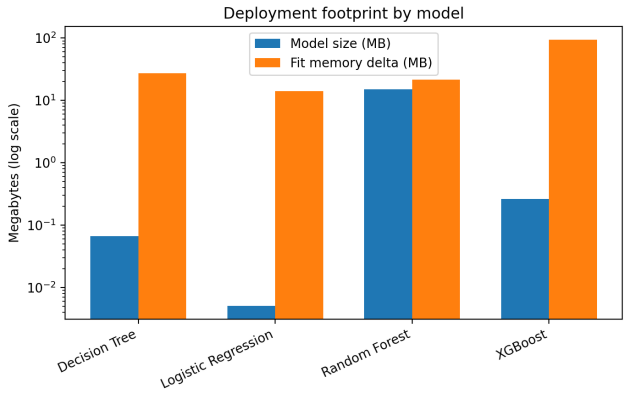

Figure 4: Deployment footprint by model. The y-axis is log-scaled to show both small serialized models and larger memory deltas.

deployment-specific cost-benefit analysis rather than by F1-score alone.

4.5 Feature importance and limits

The feature-importance results should be interpreted cautiously. In the Random Forest model, `Number` and `HTTPS` ranked highly in both impurity and permutation importance. That may reflect meaningful traffic-structure information, but it may also reflect dataset-specific shortcuts. For that reason, we do not use feature importance as causal evidence. The ablation analysis in Table 5 provides a more direct check: after dropping `Number` and `HTTPS`, all models lost some F1, but the losses were small and the tree-based models still outperformed Logistic Regression. This reduces, but does not eliminate, the concern that the result depends on those two features.

The study has several limits. First, it uses one benchmark dataset and a binary task, so it does not establish performance on every IoT environment, attack family, or multiclass formulation. The near-ceiling scores in prior CICIoT2023 work and in this paper suggest that binary detection is easier than attack-family or 33-class classification. Second, the main outer runs are attack-resampling runs rather than fully independent datasets because benign rows are reused; the disjoint sensitivity analysis addresses this limitation only at a smaller 100,000-per-class scale. Third, the supplementary imbalanced validation and disjoint sensitivity checks use stratified samples rather than a live traffic stream. Fourth,

timing and memory measurements come from notebook execution on a Windows machine, not from a dedicated edge device or resource-capped gateway. These limits do not invalidate the comparison, but they define its scope: the results are best read as a controlled, offline benchmark of model-family behavior, not as a full deployment certification.

5 Conclusion

Across five attack-resampling outer runs and 25 balanced trials per model, all three tree-based classifiers outperformed Logistic Regression on binary CICIoT2023 intrusion detection. The original design reused benign rows across outer runs, so the main paired tests should be interpreted cautiously. A disjoint sensitivity analysis that resampled both benign and attack rows preserved the same general pattern, with XGBoost achieving the highest default-threshold F1-score and Logistic Regression remaining lower. A feature-ablation check dropping `Number` and `HTTPS` caused only small F1 decreases and did not remove the tree-based advantage.

The practical lesson is not that one model is always best. XGBoost was the strongest overall option in this controlled benchmark because it combined the best default F1-score with a much smaller footprint and lower latency than Random Forest. Random Forest was competitive, especially after threshold tuning, but its size and inference time make it less attractive for constrained settings unless the extra cost is acceptable. Decision Tree was compact and nonlinear, while Logistic Regression remained the smallest and simplest baseline. For IoT intrusion detection, small F1 improvements should be judged alongside missed-attack counts, per-record latency, model size, and the hardware where the model would actually run.

Code and Data Availability

The reproducibility materials are included with this revision package: executed Python notebook, exported result tables, reviewer sensitivity tables, figures, and metadata file. The original CICIoT2023 dataset is publicly available from the Canadian Institute for Cybersecurity [5]. A public repository can be added after review if needed.

References

[1] Kayode S. Adewole, Andreas Jacobsson, and Paul Davidsson. 2025. Intrusion Detection Framework for Internet of Things with Rule Induction for Model Explanation. *Sensors* 25, 6 (2025), 1845. https://doi.org/10.3390/s25061845

[2] Salah A. Almahaqeri, Mohammed Hashem Almourish, Adel A. Nasser, Abed Saif Ahmed Alghawli, Amani A. K. Elsayed, et al. 2026. An Optimized Gradient Boosting Framework for IoT Intrusion Detection: A Comprehensive Evaluation on the CICIoT2023 Dataset. *Scientific Reports* 16 (2026), 16909. https://doi.org/10.1038/s41598-026-47399-5

[3] Zainab Alwaisi et al. 2024. Securing Constrained IoT Systems: A Lightweight Machine Learning Approach for Anomaly Detection and Prevention. *Internet of Things* 28 (2024), 101398. https://doi.org/10.1016/j.iot.2024.101398

[4] Mohammed Berhili et al. 2024. Intrusion Detection Systems in IoT Based on Machine Learning: A State of the Art. *Procedia Computer Science* 251 (2024), 99–107. https://doi.org/10.1016/j.procs.2024.11.089

[5] Canadian Institute for Cybersecurity, University of New Brunswick. 2026. IoT Dataset 2023. https://www.unb.ca/cic/datasets/iotdataset-2023.html Accessed 3 July 2026.

[6] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[7] Eric Gyamfi and Anca Jurcut. 2022. Intrusion Detection in Internet of Things Systems: A Review on Design Approaches Leveraging Multi-Access Edge Computing, Machine Learning, and Datasets. *Sensors* 22, 10 (2022), 3744. https://doi.org/10.3390/s22103744

[8] Brunel Rolack Kikissagbe and Meddi Adda. 2024. Machine Learning-Based Intrusion Detection Methods in IoT Systems: A Comprehensive Review. *Electronics* 13, 18 (2024), 3601. https://doi.org/10.3390/electronics13183601

[9] Euclides Carlos Pinto Neto et al. 2023. CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors* 23, 13 (2023), 5941. https://doi.org/10.3390/s23135941

[10] Souradip Roy, Juan Li, Bong-Jin Choi, and Yan Bai. 2022. A Lightweight Supervised Intrusion Detection Mechanism for IoT Networks. *Future Generation Computer Systems* 127 (2022), 276–285. https://doi.org/10.1016/j.future.2021.09.027

[11] İsa Yiltirak and Ali Öztürk. 2025. Comparison of Performances of Machine Learning Algorithms in Detection of Internet of Things Attacks. *Journal of Intelligent Systems: Theory and Applications* 8, 2 (2025), 130–140. https://doi.org/10.38016/jista.1635809