A Gradient Boosting Framework with Phi-K Correlation-Based Feature Selection for Intrusion Detection in IoT Networks

Emily (Yiming) Ma

The Lawrenceville School, Lawrence Township, NJ, USA

yiming760824@gmail.com

Abstract

Internet of Things (IoT) devices generate vast network traffic but can hardly run heavyweight intrusion detection systems due to resource constraints. Moreover, the signature-based detectors often fail against zero-day attacks. This study proposes a lightweight detection framework that combines Phi-K correlation-based feature selection with gradient boosting. By using the RT-IoT2022 dataset of over 120,000 real IoT traffic flows, I reduced 82 candidate features to 34 (a 58.5% reduction) and compared nine models spanning logistic regression, decision trees, bagging, random forests, and XGBoost. The tree-based ensembles clustered at the top of the comparison, with no meaningful difference between them; the tuned XGBoost model achieved 99.87% accuracy, 99.92% recall, 99.93% precision, and an F1-score of 0.9993 on held-out test data, with a false-negative rate of 0.08% and a false-positive rate of 0.64%. The feature-importance analysis shows that timing features are dominant: maximum flow inter-arrival time (40.9%) ranks first, and maximum forward inter-arrival time (27.7%) ranks second, which suggests that a small set of timing features can effectively support accurate, resource-efficient detection at IoT gateways. Limitations include a single-dataset evaluation and bias from random splitting, and I also address the outline of extensions for multi-class classification and strategies to improve robustness.

Keywords: intrusion detection; Internet of Things; XGBoost; gradient boosting; Phi-K correlation; explainable AI; network security; machine learning

1. Introduction

Internet of Things (IoT) connects numerous and various devices, from smart thermostats and medical monitors to industrial control systems and autonomous vehicles. By 2030, an estimated 29 billion IoT devices (Top Digital Forensic Tools, 2025) will be connected, generating unprecedented network traffic and a vast attack surface (Statista, 2023). The 2016 Mirai botnet attack demonstrated the catastrophic potential of IoT vulnerabilities, hijacking hundreds of thousands of compromised devices to launch distributed denial-of-service (DDoS) attacks (Vulnerability Alert, 2025) that disrupted major internet infrastructure in the United States and Europe (Antonakakis et al., 2017).

Traditional intrusion detection systems (IDS) operate by matching signatures and they compare network traffic against known attack patterns. While these systems are effective at identifying established threats, they cannot detect zero-day or novel attacks (Liao et al., 2013). This limitation is significant in today's rapidly evolving threat landscape. Machine learning provides an alternative approach: learning complex patterns from data and generalizing to identify previously unseen attacks. However, many IoT devices cannot support computationally intensive detection algorithms, so a practical IoT detector must be accurate, lightweight, and ideally interpretable enough for security practitioners to trust and act on its output.

Current IoT environments add more challenges beyond those in conventional networks. Devices are heterogeneous with different protocols—for example, MQTT, CoAP, and Zigbee, each with its distinct traffic patterns and vulnerabilities. Deployments are massive, which means the analysis must be efficient at scale. Moreover, sophisticated attackers may craft traffic specifically to bypass machine learning detectors. In these contexts, a detection framework must use effective features, remain robust across protocols, and be transparent about the signals it relies on.

The study examines whether a small, well-defined feature set can maintain detection performance. It also investigates which machine learning paradigm best suits realistic IoT traffic and which traffic characteristics matter most. The contributions are summarized in three main aspects. First, it uses Phi-K correlation coefficient, which handles mixed variable types, to select features from network traffic data, reducing the feature space by 58.5%. Second, it applies nine different models across four paradigms with systematic hyperparameter tuning and finds the tree-based ensembles statistically tied at the top, with tuned XGBoost performing as the framework's best-balanced primary model. Finally, it integrates explainable AI, showing that timing features have the greatest predictive power, and this finding has direct implications for improving efficient edge-based detection in network traffic.

2. Related Work

Network intrusion detection has developed through three generations. First-generation systems that used signature matching against known attack patterns, have achieved high precision in recognizing known threats but lack the ability to detect new or novel attacks (Liao et al., 2013). Second-generation anomaly-based systems have built statistical baselines of normal behavior and flagged deviations, and they can generally detect new attacks but tend to have high false-positive rates. This difficulty arises because it can be challenging to differentiate between malicious anomalies and legitimate but unusual traffic (Garcia-Teodoro et al., 2009). Third-generation machine learning systems learn decision boundaries from labeled samples, combining the precision of signature methods with the adaptability of anomaly detection. This study is in the range of the third generation, focusing on the feature-engineering challenge specific to IoT environments.

There are several paradigms within Machine learning approaches for intrusion detection systems. Among them, while logistic regression and linear support vector machines can provide interpretable benchmarks, they are still unable to capture complex interactions among features (Amor et al., 2004). Decision trees offer clear rule structures but are easily overfit. Ensemble methods, such as bagging, random forests, and boosting, effectively tackle these challenges by aggregating predictions from multiple decision trees (Friedman & Popescu , 2004; Breiman, 2001). One example is XGBoost, which integrates gradient-descent optimization with tree-based learning and includes built-in regularization. It has achieved great results in structured data classification (Chen and Guestrin, 2016). While deep learning architectures, including convolutional and recurrent networks, show potential promise, they demand substantial computing resources and a large amount of training data (Ferrag et al., 2020).

Feature selection is critical for intrusion detection because network traffic data are high-dimensional. Filter methods evaluate features by statistical measures that are independent of the learning algorithm. Wrapper methods evaluate subsets through model performance, and embedded methods, on the other hand, select features during the training process. Tree-based importance scores blur the line between feature selection and modeling, which motivates this study's post-hoc importance analysis. Network traffic mixes variable types — categorical protocols, continuous time measurements, and discrete packet counts — and most standard coefficients handle only one type. This comprehensive research still shows three gaps: many studies use outdated datasets, such as KDD Cup 99 and NSL-KDD, that do not accurately reflect modern IoT traffic; few apply principled feature selection methods designed for mixed-type data; and there's a lack of studies that integrate explainable AI to give practitioners actionable insight. Legacy benchmarks also predate IoT protocols like MQTT and CoAP, so models trained on them may miss traffic patterns that are prevalent in IoT deployments. This study aims to address all three gaps by evaluating a recent IoT dataset with a mixed-type selector and an interpretability analysis.

3. Methods

3.1 Dataset

This study uses the RT-IoT2022 dataset, published by Sharmila and Nagapadma through the UCI Machine Learning Repository (Sharmila and Nagapadma, 2023). It contains over 120,000 network traffic flows from three IoT device types: ThingSpeak-LED, an IoT LED controller communicating over HTTP; Wipro-Bulb, a smart bulb with wireless connectivity; and MQTT-Temp, a temperature sensor using the MQTT protocol. The dataset captures normal operation and a broad range of attacks (Table 1). The dataset was chosen for its recency (2023), device diversity, attack coverage, and adequacy for training complex models. The variable is binary: 0 indicates normal traffic (MQTT publishes, ThingSpeak, and Wipro operations) and 1 signifies any form of attack traffic (flooding, probing, credential guessing, and traffic manipulation). The

classifier must distinguish between standard machine communication and a variety of malicious behaviors (Ghazi et al., 2025). The classes are imbalanced: attack flows account for 110,610 of the dataset's 123,117 flows (89.8%), and the held-out test set was drawn by stratified sampling to preserve this proportion.

Table 1. Attack types in the RT-IoT2022 dataset.

Category	Attacks	Description
Denial of service	DDoS (Hping, Slowloris), DoS SYN	Resource-exhaustion attacks
Reconnaissance	Nmap (FIN, OS, TCP, UDP, XMAS scans)	Network mapping and vulnerability discovery
Credential attacks	SSH brute force, Metasploit	Password guessing and exploitation
Network manipulation	ARP poisoning	Man-in-the-middle attacks

3.2 Preprocessing and Phi-K Feature Selection

Preprocessing addressed data quality in two steps. First, variance analysis identified and removed features containing random or constant values, preventing noise from entering the models. Second, categorical features such as protocol and service type were converted with one-hot encoding, so that categorical variables could enter numerical algorithms without imposing artificial ordinal relationships — label encoding would wrongly imply, for instance, that one protocol is "greater than" another.

Feature selection with the Phi-K correlation coefficient (φ_K) provides consistent estimates for variable pairs regardless of type—categorical, ordinal, or interval (Baak et al., 2020). Traditional coefficients depend on the type of data: Pearson correlation assumes linear relationships between continuous variables, Spearman correlation is suitable for ordinal data, and Cramér's V is designed for categorical association (Ghazi et al., 2025). Network traffic contains all three types, so Phi-K's unified framework is an appropriate fit. For two variables X and Y, φ_K is derived from Pearson's χ^2 contingency statistic after Baak et al.'s (2020) noise correction: the statistic is corrected for the fluctuations expected under independence and calibrated against a bivariate-normal reference, so φ_K has no simple closed-form expression in χ^2 (unlike Cramér's V, which is $\sqrt{[(\chi^2/n) / \min(k-1, r-1)]}$); continuous variables are optimally binned before the table is computed.

I computed the Phi-K matrix for all feature pairs (see Figure 1). The heatmap showed three patterns: clusters with $\varphi_K > 0.9$ indicating redundant information that adds computation without improving prediction accuracy; moderate correlations among conceptually related features (for example, packet counts and aggregate header sizes), which validated data quality; and features strongly correlated with the target label. Retaining only features with $\varphi_K \geq 0.1$ against the target reduced the feature space from 82 candidate features to 34, a 58.5% reduction (refer to Table 2). The selected features span six categories (see Table 3).

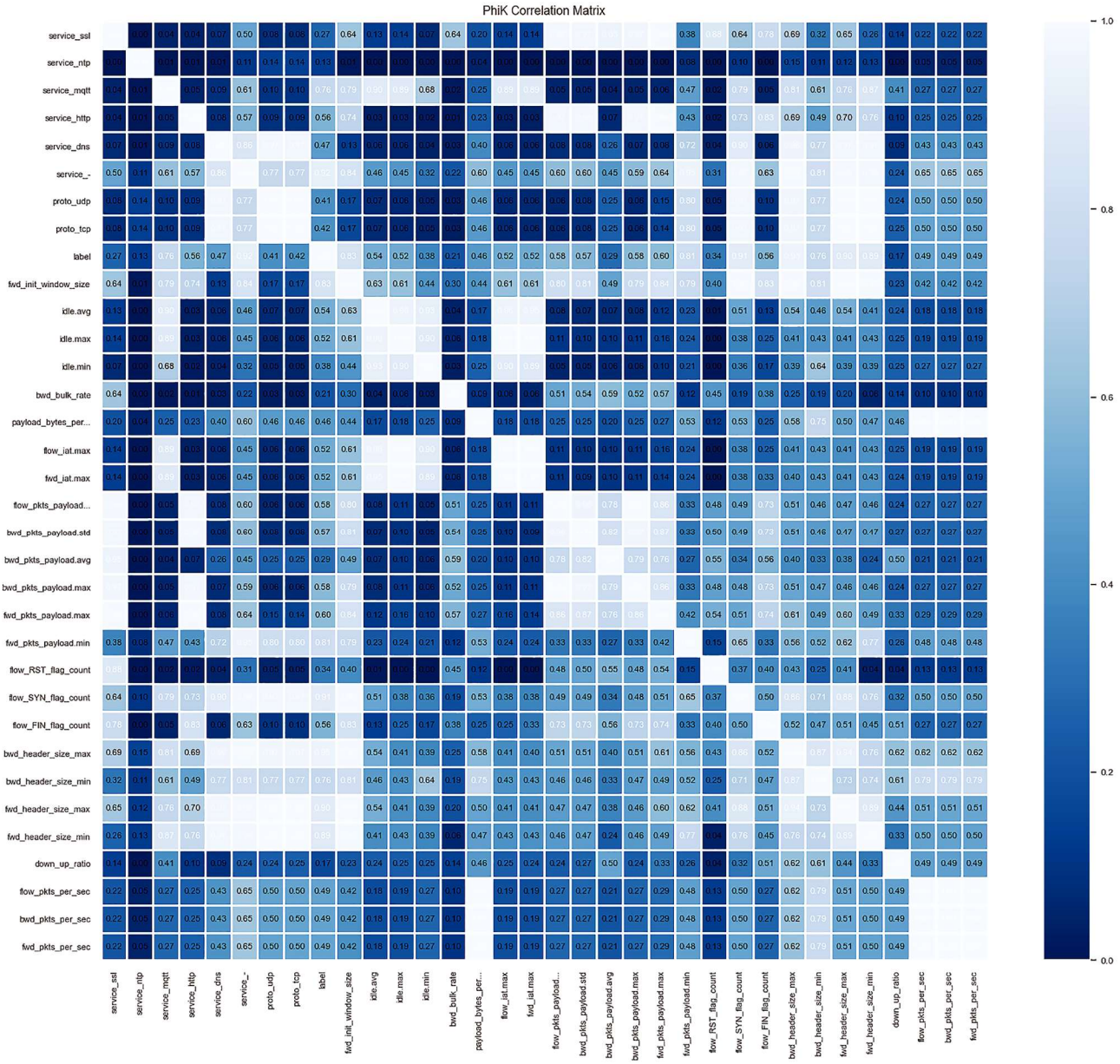


Figure 1. Phi-K correlation matrix heatmap for all feature pairs.

Table 2. Feature selection summary.

Stage	Features	Change
Original	83	—
After cleaning	82	−1.2%
After Phi-K selection	34	−58.5% vs. candidates

Table 3. Final feature categories.

Category	Description	Example features
Packet rates	Transmission frequency	Packets/s, bytes/s
Packet sizes	Dimensional statistics	Mean, maximum, and standard deviation of length
Timing patterns	Temporal features	Inter-arrival times, idle times
Protocol/service	Communication type	TCP, UDP, MQTT indicators
TCP flags	Control-flag patterns	SYN, FIN, RST counts
Header and window sizes	Packet-header and TCP window statistics	Header size maxima and minima, initial window sizes

3.3 Models

I adopted nine models across four paradigms for the evaluation. Logistic regression provided the interpretable baseline, and modeling attack probability had a logistic function of the input features. However, they were limited to linear decision boundaries. Decision trees can divide the feature space to improve class separation. They were evaluated in a standard version and a tuned version, utilizing cross-validated depth and minimum-sample constraints for evaluation. Bagging involved training multiple classifiers on bootstrap samples and then combining their predictions through majority voting. Random forests added feature randomization at each split to help decrease the correlation between trees, which further reduces variance (Breiman, 2001). Both bagging and random forests were evaluated in both standard and tuned versions.

XGBoost builds an additive model that corrects earlier errors (Chen and Guestrin, 2016), which minimizes a regularized objective $L(\varphi) = \Sigma_i\, l(y_i, \hat{y}_i) + \Sigma_k\, \Omega(f_k)$, with $\Omega(f) = \gamma T + \frac{1}{2}\lambda \Sigma_j\, w_j^2$ penalizing trees. The second-order Taylor approximation of the loss provides more informed updates. In this study, both standard and tuned versions of XGBoost were evaluated. The tuning process focused on learning rate, maximum depth, regularization parameters, and subsampling ratios. Hyperparameters for all tuned models were selected by five-fold cross-validation on the training data, optimizing the F1-score. The tuned XGBoost configuration used a learning rate of 0.1, a maximum depth of 8, $\lambda = 1$, and a subsample ratio of 0.8. For the decision tree, the search selected the unconstrained configuration (no depth limit and a minimum leaf size of one), so the tuned and standard decision trees coincide in this study.

3.4 Evaluation

I split the data into training and test subsets at an 8:2 ratio using stratified sampling to maintain class proportions. The training set contained 98,493 flows (88,487 attack and 10,006 normal) and the test set 24,624 flows (22,123 attack and 2,501 normal); a fixed random seed (42) was used for the split and for model training. Throughout model development and tuning, I kept the test set separate. For hyperparameter optimization, I used five-fold cross-validation. All models were evaluated on the same test set using the same metrics. I measured performance using accuracy, precision (the reliability of attack alerts), recall (the completeness of attack detection), the F1-score (the harmonic mean of precision and recall), and ROC-AUC (the ability to distinguish across all classification thresholds). F1-score is emphasized alongside accuracy because, in security applications, missing an attack and raising a false alarm carry asymmetric costs that raw accuracy alone does not capture. The train-test gap — the difference between training and test performance — was used as an indicator of overfitting. To enhance interpretability, I calculated feature importance using two methods: mean decrease in impurity for tree-based models and permutation importance, which assesses the drop in performance when a feature's values are randomly shuffled. I performed paired t-tests on the F1 scores obtained from five-fold cross-validation to determine whether the differences observed from the tuned XGBoost model were statistically significant.

4. Results

4.1 Model Performance Comparison

All nine models were trained on the preprocessed RT-IoT2022 data and evaluated on the held-out test set (Table 4). The tree-based ensembles form a statistically tied leading group: their test F1-scores fall between 0.9993 and 0.9996, and the paired tests in Section 4.5 find no significant difference between tuned XGBoost and the random forest or untuned XGBoost. Tuned XGBoost is retained as the primary model of this study's gradient-boosting framework: it achieved 99.87% test accuracy, 99.92% recall, 99.93% precision, and an F1-score of 0.9993, with a train–test F1 gap of +0.00050. Its appeal is balance rather than any single headline number — every metric exceeds 0.98 on both training and test data, its built-in regularization guards against overfitting, and its probability scores support threshold calibration during deployment (Section 5.2).

Two comparisons sharpen the picture. Logistic regression reached 98.7% test accuracy with an AUC of 0.9958 and a test F1-score of 0.9926, a strong baseline showing that attack and normal traffic are largely separable in the Phi-K-selected feature space; the tree ensembles nonetheless capture the remaining nonlinear structure, holding test accuracies between 0.9987 and 0.9992. The standard and tuned decision trees coincide in this study — the tuned search selected the unconstrained tree — and reached a test accuracy of 0.9989 with a recall of 0.9994, missing 13 of the 22,123 attacks in the test set. Tuning changed little anywhere: within the bagging, random-forest, and XGBoost families, tuned and standard versions differ by at most 0.001 in test F1, so the Phi-K feature space, rather than hyperparameter search, accounts for most of the performance. The two ensemble families behaved similarly: bagging and random forests reached 0.9989 and

0.9991 test accuracy in their standard forms, and their tuned versions were effectively unchanged (0.9989 and 0.9992), consistent with variance reduction through aggregation rather than any single decisive modeling choice.

Table 4. Model training and test performance.

Model	Train accuracy	Test accuracy	Train recall	Test recall	Train precision	Test precision	Train F1	Test F1	Train–test F1 gap
Logistic regression	0.98722	0.98676	0.99139	0.99096	0.99437	0.99429	0.99288	0.99262	0.00026
Decision tree	0.99981	0.99886	0.99986	0.99941	0.99992	0.99932	0.99989	0.99937	0.00052
Tuned decision tree	0.99981	0.99886	0.99986	0.99941	0.99992	0.99932	0.99989	0.99937	0.00052
Bagging classifier	0.99972	0.99886	0.99980	0.99932	0.99989	0.99941	0.99984	0.99937	0.00047
Tuned bagging classifier	0.99981	0.99890	0.99986	0.99928	0.99992	0.99950	0.99989	0.99939	0.00050
Random forest	0.99981	0.99915	0.99986	0.99955	0.99992	0.99950	0.99989	0.99953	0.00036
Tuned random forest	0.99981	0.99923	0.99986	0.99964	0.99992	0.99950	0.99989	0.99957	0.00032
XGBoost	0.99963	0.99907	0.99984	0.99941	0.99975	0.99955	0.99980	0.99948	0.00032
Tuned XGBoost	0.99954	0.99866	0.99982	0.99923	0.99967	0.99928	0.99975	0.99925	0.00050

4.2 ROC Analysis

Figure 2 displays the ROC curve for the logistic regression model. The curve approaches the upper-left corner of the graph, and its AUC of 0.9958 indicates strong discrimination capability, demonstrating that effective feature selection alone can make even a linear model nearly perfectly discriminative on this dataset. The other models ranked still higher — test AUCs from 0.9967 for the decision trees to above 0.9999 for the tuned random forest and the XGBoost variants — so the logistic-regression curve is best read as a lower bound on the ranking quality available in this feature space. The curve also shows the trade-off at different operating points: moving the decision threshold towards higher recall captures more attacks but also increases the number of false alarms. Threshold calibration during deployment (as discussed in Section 5.2) can therefore be viewed as a policy choice rather than merely a modeling issue.

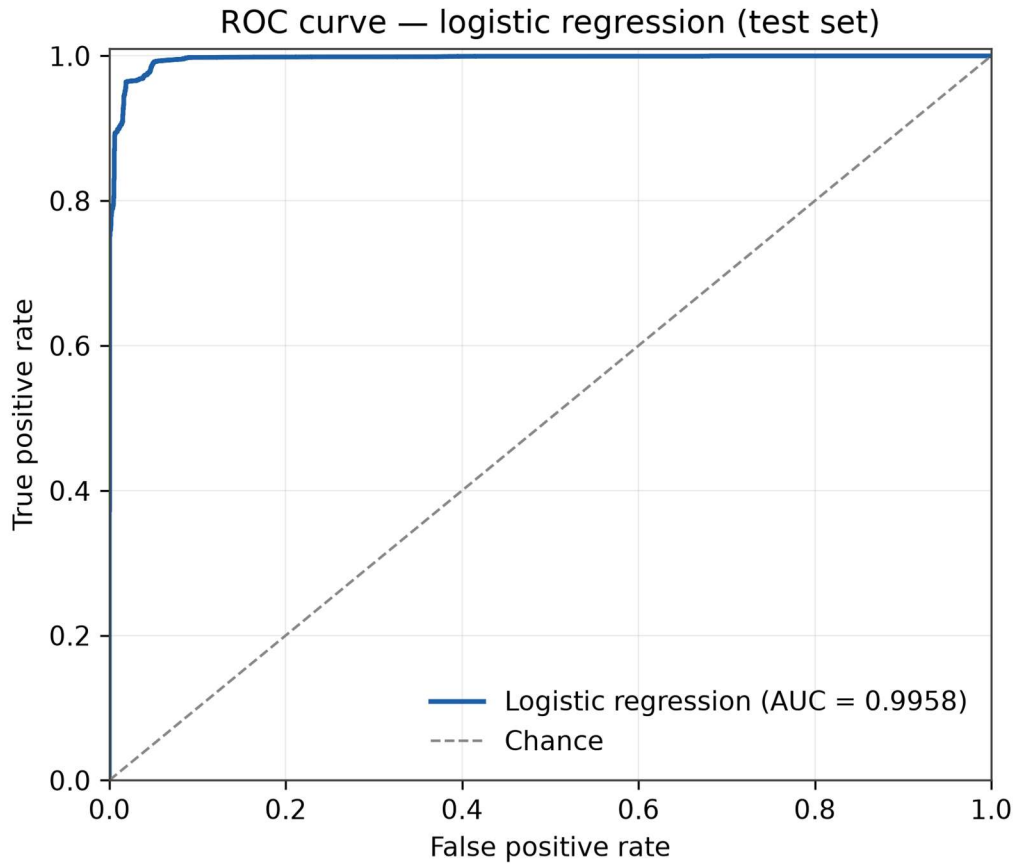


Figure 2. ROC curve for the logistic regression model (AUC = 0.9958).

4.3 Confusion Matrix Analysis

Figure 3 shows the confusion matrix for the tuned XGBoost model on the test data. The true-positive rate was 99.92% and the true-negative rate 99.36%, giving a false-positive rate of 0.64% and a false-negative rate of 0.08% — 2,485 true negatives, 16 false positives, 17 false negatives, and 22,106 true positives among the 24,624 test flows. The low false-negative rate matters most for security: a missed attack can mean a data breach and system compromise. The low false-positive rate matters for operations: it keeps analysts' attention from being wasted on false alarms.

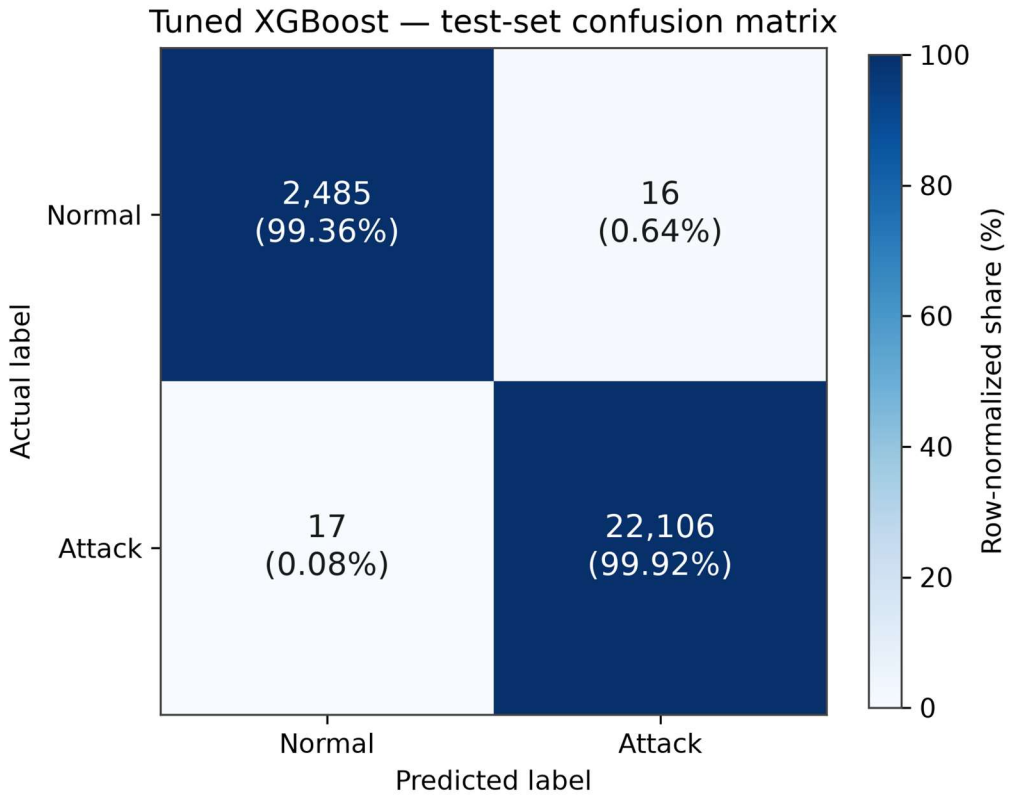


Figure 3. Confusion matrix for the tuned XGBoost model on the test data.

4.4 Feature Importance Analysis

Explainable AI analysis of the tuned XGBoost model produced consistent patterns across importance measures (Figure 4, Table 5). Timing features dominate: maximum flow inter-arrival time ranks first at 40.9% of total gain, and maximum forward inter-arrival time ranks second at 27.7%; together the two inter-arrival maxima account for 68.7% of the model's gain. The timing result is interpretable through attack behavior. DoS and DDoS flooding produces very short inter-arrival times; reconnaissance scans show irregular, longer gaps between probes; normal IoT traffic follows regular, predictable device polling intervals. Secondary signals reinforce the picture: the standard deviation of forward packet payload size (7.4%) and the initial backward window size (6.4%) add size and handshake evidence, while size and rate features further down the ranking capture the volume and burstiness that distinguish flooding from ordinary transfers. Permutation importance corroborates the leading feature — shuffling maximum flow inter-arrival time produces the largest drop in test F1 (0.037) — and ranks minimum forward packet payload size second (0.031), so the two views agree that a small set of timing and size features carries the detection signal. The concentration of predictive power in timing features has a practical consequence: a lightweight classifier using only the top few features could plausibly achieve high detection rates with minimal computation, enabling deployment on resource-constrained IoT

gateways. That reduced-feature classifier was not itself tested here, so this remains an implication of the importance ranking, not a measured result.

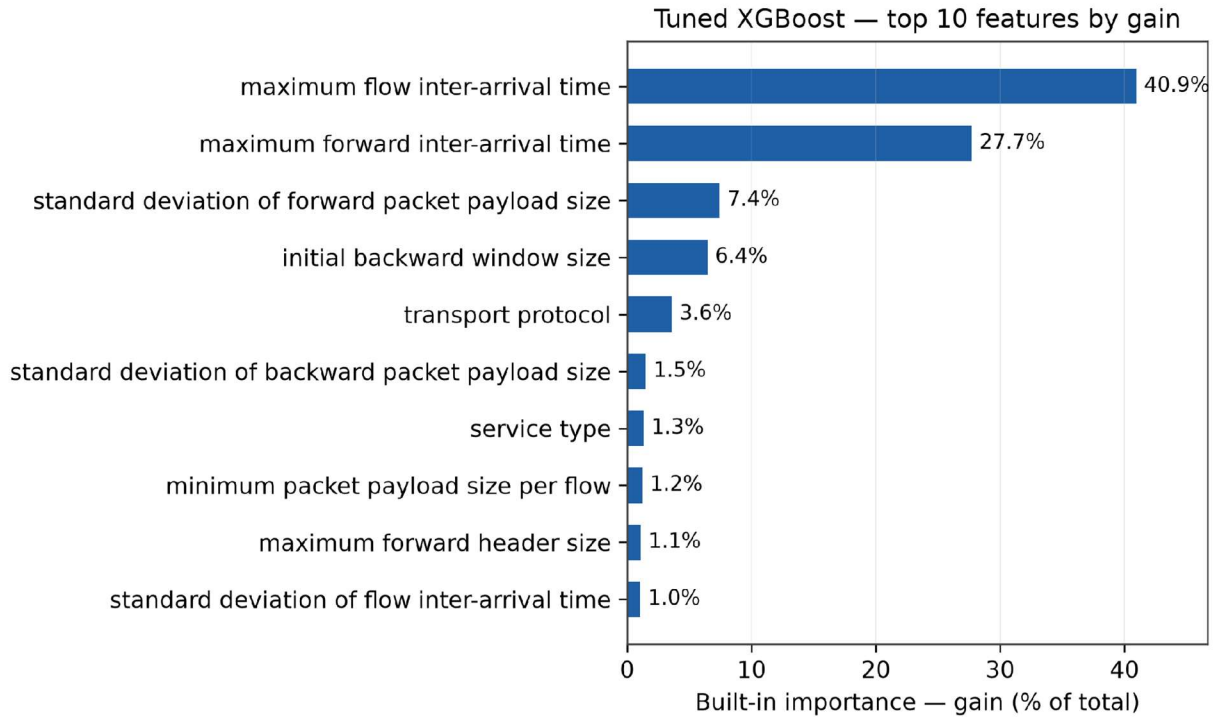


Figure 4. Top 10 features by importance from the tuned XGBoost model.

Table 5. Top 10 most important features (tuned XGBoost).

Rank	Feature	Importance	Category
1	Maximum flow inter-arrival time	0.409	Timing patterns
2	Maximum forward inter-arrival time	0.277	Timing patterns
3	Standard deviation of forward packet payload size	0.074	Packet sizes
4	Initial backward window size	0.064	Header and window sizes
5	Transport protocol	0.036	Protocol/service
6	Standard deviation of backward packet payload size	0.015	Packet sizes

7	Service type	0.013	Protocol/service
8	Minimum packet payload size per flow	0.012	Packet sizes
9	Maximum forward header size	0.011	Header and window sizes
10	Standard deviation of flow inter-arrival time	0.010	Timing patterns

4.5 Statistical Significance

Paired t-tests on five-fold cross-validation F1-scores confirmed that the tuned XGBoost model significantly outperformed logistic regression (mean ΔF1 +0.0068, $p < 0.001$). Its margin over the tuned decision tree was statistically significant but negligible in size (mean ΔF1 +0.0002, $p = 0.032$), and differences from the random forest (mean ΔF1 ≈ 0.000, $p = 0.455$) and untuned XGBoost (mean ΔF1 ≈ 0.000, $p = 0.401$) were not statistically significant. The leading ensembles are therefore best read as a tied group, and the choice of tuned XGBoost as the primary model rests on its balance of performance, regularization, and probability outputs rather than on a significant margin over the other ensembles.

5. Discussion

5.1 Summary of Findings

The analysis yielded three key findings. First, Phi-K feature selection reduced dimensionality by 58.5% while every one of the nine models reached a test F1-score of at least 0.9926, enabling efficient deployment. Second, the tree-based ensembles form a statistically tied leading group (test F1-scores 0.9993–0.9996); within it, the tuned XGBoost model provided the best balance between performance and generalization for this study's framework, achieving 99.87% accuracy, an F1 score of 0.9993, and a train–test F1 gap of +0.00050. Third, timing features, especially the maximum inter-arrival times, contained the most discriminative information. This finding gives us an important priority: to measure timing first. Finally, hyperparameter tuning changed performance only marginally — within each family, tuned and standard versions differ by at most 0.001 in test F1 — which suggests that the Phi-K-selected feature space, more than tuning, is what makes this traffic so separable, and that regularized models remain reliable on unseen traffic.

5.2 Practical Implications

For general deployment, tuned XGBoost is the recommended model; for resource-constrained settings, a single decision tree offers comparably strong performance (test F1-score 0.9994) with a much simpler structure. Implementations should prioritize timing features (inter-arrival times) and flow statistics (packet counts) to gain the most accuracy per unit of feature computation, and detection thresholds can be calibrated from the ROC analysis — higher sensitivity for critical systems, balanced thresholds for general use. To maintain effectiveness against evolving attack techniques, models should be retrained using recent traffic data. The dimensionality reduction allows a tiered architecture: a gateway can first screen traffic with a lightweight model on a few timing features and then forward only suspicious flows to the cloud for full analysis. When staff resources for alert management are limited, the low false-positive rate of 0.64% makes a strong case for deployment. It ensures that the daily volume of alerts remains manageable, allowing a small team to investigate every alert instead of just sampling them.

5.3 Comparison with Prior Work

Earlier research on legacy data indicated lower accuracy, especially in the absence of mixed-type feature selection. For instance, Ahmad et al. (2021), in their systematic review of the field, report random-forest accuracy of 94.3% on the NSL-KDD dataset with information-gain selection in the studies they surveyed, but without any interpretability analysis. This study improves upon that baseline in three key areas: it achieves higher accuracy on a newer dataset specific to IoT, employs a principled mixed-type feature selector, and includes an integrated interpretability analysis.

5.4 Limitations

Several limitations should be acknowledged. First, results are specific to one dataset: performance may vary on other IoT device types or on attack patterns absent from the training data. Second, my evaluation used a random 80/20 split of traffic flows. Flows generated by the same device during the same attack burst are highly correlated, so near-duplicate flows can appear on both sides of the split and inflate test scores; this may partly explain the near-perfect scores of several models, including the unpruned decision tree (test accuracy 0.9989). A group-based split by device or capture session would give a more conservative estimate of generalization, and I recommend it as a robustness check for future work. Additionally, the task here is binary (Normal vs. Attack); distinguishing attack types remain untested. Third, attack techniques evolve, so model validity will diminish over time. Fourth, no evaluation against adversarial examples specifically designed to evade detection was performed. Fifth, no detailed inference-time or memory analysis for edge deployment was conducted, so the lightweight claim rests on the reduced feature count.

6. Conclusion and Future Work

This study presents a machine learning framework for IoT intrusion detection built on three contributions: Phi-K feature selection that cuts dimensionality by 58.5% while preserving discriminative information; a systematic comparison of nine models identifying tuned XGBoost

as a co-leading, best-balanced model at 99.87% accuracy, 99.92% recall, and 99.93% precision; and an explainable AI analysis identifying maximum flow inter-arrival time (40.9% importance) as the leading predictor, with direct implications for lightweight detection design. The results show that carefully designed machine learning can achieve near-perfect intrusion detection on IoT traffic while remaining interpretable for security-critical use. As IoT devices continue to spread through homes, medical systems, and critical infrastructure, detectors of this kind — accurate, economical with features, and explainable — become part of the basic protection of the digital ecosystem and the people who depend on it.

Promising directions include multi-class classification to enable targeted responses to specific attack types; evaluation and improvement of adversarial robustness; adaptation to real-time streaming with incremental learning; federated learning for collaborative training across IoT deployments without centralizing sensitive traffic; and, as an extension of my ongoing IoT-based fall-detection research for elderly safety, a unified anomaly-detection framework that uses shared IoT sensor infrastructure to recognize both network intrusions and physical safety events.

Acknowledgments

This research was conducted as part of the Stanford Pre-Collegiate Summer Institutes program. I thank the program instructors for their guidance and feedback. The RT-IoT2022 dataset was provided by Sharmila and Nagapadma through the UCI Machine Learning Repository.

References

Ahmad, I., et al. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32(1), e4150.

Amor, N. B., et al. (2004). Naive Bayes vs decision trees in intrusion detection systems. Proceedings of the 2004 ACM Symposium on Applied Computing, 420–424.

Antonakakis, M., et al. (2017). Understanding the Mirai botnet. 26th USENIX Security Symposium, 1093–1110.

Baak, M., et al. (2020). A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics. Computational Statistics & Data Analysis, 152, 107043.

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

Chen, T., and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

Ferrag, M. A., et al. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications, 50, 102419.

Friedman, J. H., & Popescu, B. (2004). Ensemble learning for prediction. https://dl.acm.org/citation.cfm?id=1037421

Ghazi, Y., Tabaa, M., Ennaji, M., & Zaz, G. (2025). An Explainable Markov Chain–Machine Learning Sequential-Aware Anomaly Detection Framework for Industrial IoT Systems Based on OPC UA. Sensors, 25(19), 6122. https://doi.org/10.3390/s25196122

Garcia-Teodoro, P., et al. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1–2), 18–28.

Liao, H.-J., et al. (2013). Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications, 36(1), 16–24.

Sharmila, B. S., and Nagapadma, R. (2023). RT-IoT2022. UCI Machine Learning Repository. https://doi.org/10.24432/C5P338

Statista. (2023). Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030. Statista Research Department.

Top Digital Forensic Tools: Unveiling the Arsenal of Digital Forensic Investigators in 2025 - CYFOR Forensics. https://cyfor.co.uk/top-digital-forensic-tools-unveiling-the-arsenal-of-digital-forensic-investigators-in-2025/

Vulnerability Alert: KEVs, Botnet Activity, and APT Campaigns | Weekly Reports | Loginsoft. https://www.loginsoft.com/reports/weekly/vulnerability-alert-kevs-botnet-activity-and-apt-campaigns