JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 11 Issue 01
2026

Research Article
https://doi.org/10.56979/1101/2026/1352

Comparative Analysis of Machine Learning Classifiers for Detecting Backdoor Malware in IoT Networks

Adeeb Alsaaidah[1*]

[1]Department of Information Systems and Network, The World Islamic Sciences and Education University, Amman, Jordan.
[*]Corresponding Author: Adeeb Alsaaidah. Email: adeeb.saaidah@wise.edu.jo

Received: January 05, 2026 **Accepted:** April 20, 2026

Abstract: The growth of Internet of Things (IoT) networks has led to the fact that it can now be more exposed to cyber-attacks, including those based on backdoor malware that makes it possible to gain unauthorized and persistent access to the compromised devices. Proper and effective identification of such attacks is hence critical in ensuring the safety of the IoT environments. This paper provides a comparative overview of classical machine-learning (ML) models of binary backdoor malware detection in IoT networks, in which the network traffic samples are either labeled Backdoor malware or Benign. The experimental workflow starts with an unbalanced backdoor of CICIoT2023 dataset, which is then preprocessed and balanced to create the final benchmark dataset to be used in model evaluation. This dataset is composed of 39,160 samples and 47 features, and the distribution of classes are the equal (19,580 benign and 19,580 backdoor malware samples). The ML pipeline was applied with a single evaluation pipeline based on data cleaning, label encoding, and feature standardization, and then 10-fold cross-validation. Twenty classical ML classifiers were benchmarked, including ensemble, linear, probabilistic, kernel-based, instance-based, and neural-network-based ML methods. Accuracy, Precision, Recall, F1-score and execution time were used to measure performance indicates. The findings indicate that tree-based ensemble models always perform better than other models with ExtraTrees having the highest overall performance (Accuracy = 0.999872, Precision = 0.999872, Recall = 0.999872, F1-score = 0.999872), then followed by RandomForest (0.999719) Among the best performing models, XGBoost also produced very competitive results (Accuracy = 0.999489) with desirable computational efficiency. These results emphasize the use of ensemble-based ML methods as highly efficient and feasible baselines in detecting backdoor malware in IoT and the evolution of effective, deployment-friendly IoT security tools.

Keywords: Internet of Things (IoT); Backdoor malware detection; Machine learning; CICIoT2023 dataset

1. Introduction

The Internet of Things (IoT) has revolutionized the current computing by allowing sensors, smart appliances, industrial controllers, medical devices, and smart infrastructure to connect on a large scale [1, 2]. This ubiquitous integration has enhanced automation, tracking, and decision-making in most sectors, but it has also come with significant security threats [3]. IoT devices often run on limited resources, are heterogeneously configured, and are not hardened against security threats, thus becoming vulnerable to cyberattacks [4, 5].

Attacks on IoT ecosystems have evolved fast, using weak credentials, unsecure services, old firmware, and unprotected communications interfaces to attack devices at scale [6, 7].

Backdoor malware is one of the most dangerous types of IoT malware as it was created to create and support unauthorized access to hacked systems [8, 9]. In contrast to other attacks that are mostly aimed at causing disruption in the short-term, backdoor malware is often designed to establish persistent control channels via which attackers can remotely give orders, deploy other payloads, steal information, or orchestrate multi-stage malicious operations [10]. This persistence is particularly harmful in IoT contexts since devices with compromised security can continue to operate and appear normal but secretly engage in malicious activity. An IoT device that is infected with a backdoor can be used in the construction of botnets, lateral movement in local networks, manipulation of traffic, surveillance and can be used as a launch pad to larger campaigns, including distributed Denial-of-Service (DDoS) [11-13]. Improving backdoor malware in IoT networks is difficult because of several factors. First, the traffic pattern of IoT is very diverse by virtue of the type of devices, protocols and manner of usage. Second, malicious activity can be intermittent or hidden in a legitimate flow of communication, and the use of static thresholds and rule-based signatures is not sufficiently effective in most scenarios. Third, the amount of network traffic produced in IoT ecosystems can be high, which necessitates detection methods that are both precise and computationally efficient [14]. Such difficulties encourage the application of machine learning (ML), which can discover discriminative patterns given by data and helps to detect malicious behavior in an adaptive manner.

The deep learning has gained considerable focus in the context of cybersecurity research [15, 16], classical ML frameworks are also of great relevance in the context of IoT malware detection [17-19], especially when it comes to deployment-related applications. Classical ML classifiers tend to be more predictive and less computationally expensive, have less complex training pipelines, and are easier to reproducible. ML classifiers also work well with tabular network-traffic descriptions and can be compared systematically within a framework that is comparable across experiments. Nevertheless, the results of the model can differ significantly regarding preprocessing strategy, class distribution, feature properties, and validation design [20]. Thus, more stringent comparative research should be conducted to determine which pairs of ML classifiers provide the most reliable and feasible performance to a particular type of IoT attacks, such as backdoor malware.

Therefore, this paper performs a comparative study of classical ML classifiers in the detection of backdoor malware within the IoT networks. The detection task is presented as a binary classification problem, in which the instances of traffic are classified as Backdoor Malware or Benign. This analysis starts with an unbalanced dataset and uses pre-processing and balancing operations to generate a benchmark-friendly dataset to be used to test the model. This is followed by experimental to evaluate 20 classical ML classifiers with 10-fold cross-validation. The assessed models range a wide range of learning paradigms, such as ensemble, linear, probabilistic, kernel-based, instance-based, and neural-network-based ML methods. Accuracy, Precision, Recall, F1-score and the execution time are used to measure performance to allow both predictive and computational comparisons.

The significance of this research is not merely in the reporting of high detection but also offers a methodologically stable standard of binary backdoor malware detection in IoT-based environments. The work provides useful insights into the relevance of models to the data and analytics of IoT security using the actual implemented classical ML pipeline and a well-defined balanced dataset. The findings indicate that tree-based ensemble models perform particularly well in this task, with almost perfect discrimination and high practical utility, thus creating solid foundations of future IoT malware detectors.

Contributions of this work are summaries as follows:

1. Focused backdoor benchmark in IoT traffic: A binary detection benchmark for Backdoor_Malware vs Benign traffic in IoT traffic, developed by the backdoor subset of CICIoT2023 with an explicitly documented preprocessing and balancing workflow, that results in a reproducible evaluation dataset.
2. Broad classical ML comparison under one protocol: A single experiment comparison of 20 classical machine learning classifiers across ensembles, boosting frameworks, linear/online learners,

kernel/instance-based methods, probabilistic models, discriminant analysis, and shallow MLP under the same validation protocol.

3. Error level analysis for operational relevance: Confusion matrices for each classifier are included for interpretation of false alarms vs missed detections, critical for practical IoT security monitoring.
4. Guidance for model selection: A clear and concise direction on how to select models for deployment in IoT is provided by the benchmark, specifically regarding the selection of the most effective/efficient models .

The remainder of this paper is organized as follows. Section 2 reviews related work on backdoor malware detection in IoT environments. The materials and methods section 3 describe the dataset and preprocessing and balancing workflow and machine learning models and validation strategy and performance metrics. The experimental results are reported and discussed in Section 4 through quantitative comparisons and confusion-matrix analysis and runtime evaluation. Section 5 concludes the paper by presenting research directions that will enhance IoT backdoor detection systems for practical use in future studies.

2. Related Work

Backdoor malware has emerged as a consistent threat in both consumer and industrial IoT ecosystems because it can be used to create stealthy, long-lasting unauthorized access and to facilitate multi-stage attacks (e.g., data exfiltration, lateral movement, and service disruption). Therefore, recent literature has investigated the backdoor detection with machine learning (ML) in various settings (malware binaries, Industrial IoT, and SCADA/ICS networks) with an increasing focus on such practical considerations as class imbalance, feature reduction, and runtime efficiency. The most relevant studies (including the attached papers) are reviewed in the following paragraphs and place the motivation of the present benchmark study.

Khaliq et al. [21] explored the behavioral analysis view of backdoor malware and studied samples with heap overflow vulnerabilities. They used their pipeline to combine static and dynamic analysis to extract features, feature engineering and classification with J48, Naive Bayes and Simple Logistic. The authors confirmed their approach with train/test split and 10-fold validation, obtaining around 90.29% accuracy (train/test) and 84.46% accuracy (10-fold validation) which illustrates the utility of behavior-driven ML to identify patterns of backdoor exploitation.

Khan et al. [22] tested backdoor detectors based on ML models in the field of Industrial IoT and explicitly investigated the impact of dataset balancing and feature reduction. They compared a few ML models using the CCCS-CIC-AndMal-2020 dataset and demonstrated that the Random Forest model has the best performance of 99.98 percent accuracy in cases when the dataset is balanced with the help of SMOTE. Notably, they also showed that training and testing time may be reduced by a factor of 50 with a minimal reduction in predictive accuracy when strongly correlated features are removed and they are trained in a reduced feature space, which is important in terms of computational efficiency as a key deployment factor.

Dash et al. [23]compared LightGBM to other algorithms (Random Forest, SVM, XGBoost and Neural Networks) in the detection of backdoor malware in SCADA/critical-infrastructure environments. Their work has focused on assessing both predictive metrics and training time and concluded that LightGBM provided high overall detection results and good execution properties, which justifies its application in real-time detection conditions with the operational technology setting. (2024_2)

Panda et al. [24] studied the IoT resilience to backdoor malware (along with DNS spoofing) more recently and compared classical models such as Random Forest, Decision Tree, and Logistic Regression, with the strongest results in their experiments reported with the Random Forest. This paper supports an apparent finding in the literature: ensemble learners often offer robust baselines on which to detect IoT threats, even in the context where the problem environment is broader than a single type of attack.

Altogether, the literature supports the applicability of ML to the detection of backdoor malware in malware behavior analysis, Industrial IoT and SCADA contexts, as well as highlighting the role of class imbalance and computational efficiency. Nonetheless, the literature does not provide a reproducible standard that concentrates on binary backdoor detection of contemporary IoT traffic, compares a wide array of classical ML classifiers with the same preprocessing and validation, and presents both runtime and predictive measures. To

fill this gap, the current study presents 20 classical ML Backdoor vs Benign detector benchmarks on a balanced dataset generated by CICIoT2023 based on 10-fold cross-validation and supplemented by confusion matrices and comparative plots of metrics to facilitate a practical and deployment-oriented model selection.

3. Materials and Methods

This section describes the experimentation approach to binary backdoor malware detection in IoT networks, such as the dataset, preprocessing and balancing, the evaluated machine learning classifiers, and the validation protocol. The workflow is illustrated in Fig. 1, where benign and backdoor traffic is extracted out of CICIoT2023 to create the target dataset, then a systematic preprocessing phase (cleaning, encoding/transformations, normalization, and class-imbalance handling) is performed to create a balanced benchmark set. The models are evaluated with 10-fold cross-validation and are reported in terms of Accuracy, Precision, Recall, F1-score and runtime. All the parts of this pipeline are detailed in the following sub sections.

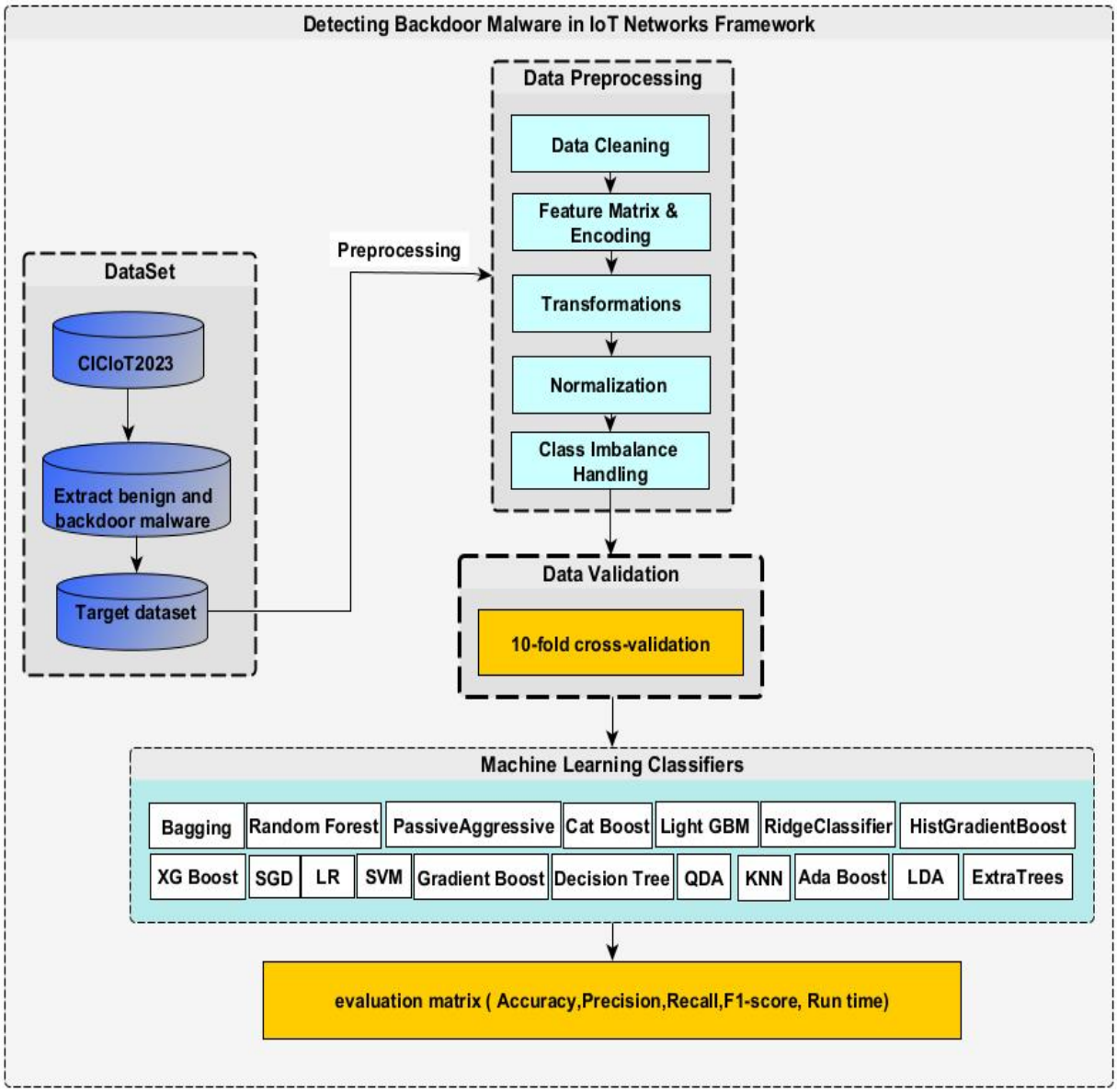

Figure 1. Backdoor malware detection workflow

3.1. Dataset Description

The experiments were performed with the help of a backdoor malware traffic dataset that is based on the CICIoT2023 IoT cybersecurity collection [25]. The dataset of the has 39,160 samples and 47 columns of data, with the final column being the target label and the rest of the columns a feature of the input. The binary target classes Backdoor Malware and Benign have equal numbers of instances in the classes with 19,580 instances per class, providing a balanced evaluation environment. This balanced formulation facilitates consistent comparison among different classifiers and reduces the possibility that high performance may be due to the dominance of majority classes as opposed to actual discriminative ability.

Within the framework of IoT security, it is worthwhile paying particular attention to backdoor malware traffic since the compromise of a backdoor can often provide a means of unauthorized access persistently and be a precursor to further malicious actions. In isolating backdoor-related traffic and benchmarking models on a binary detection task, this work seeks to offer an easy to understand, practical evaluation of the effectiveness with which classical machine learning techniques can be utilized to distinguish between malicious backdoor behavior and legitimate IoT network traffic within a controlled experimental environment. Table 1 lists the Features of CICIoT2023 dataset.

Table 1. CICIoT2023 Features list

No.	Feature Name	No.	Feature Name
1	flow_duration	24	SMTP
2	Header_Length	25	SSH
3	Protocol Type	26	IRC
4	Duration	27	TCP
5	Rate	28	UDP
6	Srate	29	DHCP
7	Drate	30	ARP
8	fin_flag_number	31	ICMP
9	syn_flag_number	32	IPv
10	rst_flag_number	33	LLC
11	psh_flag_number	34	Tot sum
12	ack_flag_number	35	Min
13	ece_flag_number	36	Max
14	cwr_flag_number	37	AVG
15	ack_count	38	Std
16	syn_count	39	Tot size
17	fin_count	40	IAT
18	urg_count	41	Number
19	rst_count	42	Magnitue
20	HTTP	43	Radius
21	HTTPS	44	Covariance
22	DNS	45	Variance
23	Telnet	46	Weight

3.2. Dataset Preprocessing and Balancing

A structured preprocessing and balancing workflow of the original dataset was used to provide data quality and reduce the bias of the classes and provide a consistent benchmarking pipeline prior to the model evaluation. The final benchmark dataset is the output of this step, with which the comparative machine learning experiments were carried out. Below are the preprocessing steps.

3.2.1. Data Cleaning and Invalid-Value Handling

The raw data was initially checked concerning the missing data and duplication of the records and invalid numeric data. Non-finite values and placeholder symbols were taken care of to prevent instability when training the model. Let x_{ij}denote the value of feature $j$for sample i. Invalid values were processed as follows:

$$x'_{ij} = \begin{cases} \text{NaN}, & \text{if } x_{ij} \in [+\infty, -\infty, ?] \\ x_{ij}, & \text{otherwise} \end{cases} \quad (1)$$

After this transformation, rows with unresolved missing values were excluded in the benchmark preparation script to ensure a clean input matrix for model comparison. This step reduces noise and prevents failures in downstream classifiers.

3.2.2. Feature Matrix and Target Extraction

The dataset was cleaned and then divided into target labels and input features. If the dataset contains m columns, the first $m-1$columns form the feature matrix X, while the final column corresponds to the target vector y:

$$X = [x_1, x_2, \dots, x_{m-1}], y = x_m \quad (2)$$

where $X \in \mathbb{R}^{n\times(m-1)}$and $y \in \mathbb{R}^n$, with $n$denoting the number of samples. In this study, the final benchmark dataset contains 46 input features and one target label column.

3.2.3. Categorical Feature Encoding

Part of the columns were categorically defined and thus had to be numerically encoded to be trained in classical ML models. The columns that were used in the implemented benchmark code were converted to label encoding. For a categorical feature $f_j$with category set $\mathcal{C}_j = [, c_2, \dots, c_k]$, the transformation is defined as:

$$\phi_j(c_t) = t-1, t = 1,2, \dots, k \quad (3)$$

Thus, each categorical value is mapped to an integer index. This conversion enables numerical processing while preserving category identity for the evaluated classifiers.

3.2.4. *Class Label Transformation*

In a similar manner, the target labels were coded into binary to facilitate the supervised learning scenario in a two-class detection task (Backdoor malware vs benign traffic).Let $y_i$denote the original class label for sample i. The encoded label $y'_i$is defined as:

$$y'_i = \begin{cases} 0, & \text{if } y_i = \text{Benign} \\ 1, & \text{if } y_i = \text{Backdoor\_Malware} \end{cases} \quad (4)$$

This representation enables the use of binary classification algorithms and standard evaluation metrics in a consistent manner.

3.2.5. Normalization

To improve numerical stability and ensure fair comparison across model families, feature standardization was applied in the benchmark pipeline using z-score normalization (via StandardScaler() within each cross-validation fold). For each feature x_j, the standardized value $z_{ij}$is computed as:

$$z_{ij} = \frac{x_{ij}-\mu_j}{\sigma_j} \quad (5)$$

where $\mu_j$and $\sigma_j$denote the mean and standard deviation of feature j, respectively, estimated from the training partition only within each fold. This step is particularly important for distance-based, linear, and margin-based classifiers, while maintaining a unified preprocessing protocol across all models.

3.2.6. Class Imbalance Handling and Dataset Balancing

The original dataset exhibited class imbalance. To mitigate bias toward the majority class and support a fair binary benchmark, a balancing procedure was applied during preprocessing to generate the balance dataset . In the preprocessing workflow, SMOTE (Synthetic Minority Over-sampling Technique) was used on the training data after a stratified split, which prevents leakage from synthetic samples into the evaluation portion.

Conceptually, for each minority-class sample x_i, a synthetic sample $\tilde{x}$is generated along the line segment between $x_i$and one of its k-nearest minority neighbors x_{nn}:

$$\tilde{x} = x_i + \lambda(x_{nn} - x_i), \lambda \in [0,1] \quad (6)$$

This process increases minority-class representation and improves class balance. After preprocessing and balancing, the final benchmark dataset used in this study contains equal numbers of Benign and Backdoor Malware instances.

3.3. Data Validation

To obtain reliable and unbiased estimates of model generalization performance, the proposed framework adopts a 10-fold cross-validation (CV) validation strategy [26], as illustrated in Fig. 1. Let the balanced dataset be denoted by $D = [(x_i, y_i)]_{i=1}^{N}$, where $x_i$is the feature vector and $y_i \in [0,1]$represents the binary class label (Benign vs Backdoor_Malware). The dataset is randomly partitioned into $K = 10$approximately equal and non-overlapping folds $\{D_1, D_2, \dots, D_{10}\}$. For each iteration k, the model is trained on the union of the remaining folds and evaluated on the held-out fold:

$$\text{Train set} = D \setminus D_k, \ \text{Test set} = D_k, \qquad k = 1, \dots, 10 \quad (7)$$

This procedure ensures that every sample is used exactly once for testing and nine times for training, reducing the risk of optimistic bias associated with a single split. In addition, cross-validation mitigates variance in performance estimation by averaging results over multiple train–test partitions. In this study, the folds were generated using KFold with shuffling and a fixed random seed ($K = 10$, shuffle = True, random_state = 42) to improve reproducibility.

To prevent pre-processing-induced leakage and maintain fairness across classifiers, classical ML models were evaluated within a unified pipeline in which feature standardization is applied during each fold. Predictions were generated using cross-validated inference, and the final reported metrics (Accuracy, Precision, Recall, F1-score, and runtime) reflect the aggregated performance across the entire dataset under the 10-fold CV setting. This validation design provides a robust and consistent basis for comparing diverse machine learning models for backdoor malware detection in IoT networks. Cross-validated predictions using 10-fold k-fold evaluation are used in the benchmark to mitigate optimistic bias. Scale-sensitive models are assessed in a pipeline where the standardization is fit on a training split of each fold and transform the test split of each fold. Furthermore, the dataset used to perform the benchmarking is class balanced and the cross-validation is not oversampled. Such design choices minimize leakage in the input data for preprocessing and prevent any test fold from being seen by the model during training.

3.4. Machine Learning Models

To compare binary backdoor malware detection in IoT networks, 20 classical ML classifiers are evaluated in this work under a same experimental environment setting. The models were chosen to encompass complementary learning paradigms typically employed in cybersecurity analytics such as ensemble learning [27-29], boosting [16], linear/online learning [30], probabilistic and discriminant classifiers [31], kernel-based learning [32, 33], instance-based learning and a shallow neural network baseline. Table 2 consists of the evaluated models sorted by family, making it possible to compare across the algorithmic paradigms instead of just an individual classifier. This benchmarking design, based on families, allows you to see clearly what learning paradigms are the most effective to identify backdoor malware in IoT traffic, and at the same time, the cost of the computational cost of each of the approaches.

Table 2. Assessed ML models by family.

Model Family	Models
Tree-based Ensembles & Bagging	Bagging, Random Forest, Extra Trees
Boosting Ensembles	AdaBoost, Gradient Boosting, HistGradientBoost
GBDT Frameworks	XGBoost, LightGBM, CatBoost
Single Tree	Decision Tree
Linear / Online Models	Logistic Regression, SGD, RidgeClassifier, PassiveAggressive

Probabilistic	GaussianNB
Kernel-based	SVM
Instance-based	KNN
Discriminant Analysis	LDA, QDA
Shallow Neural Network	MLP

3.5. Evaluation Metrics

Evaluation Metrics The effectiveness of each ML classifier was measured using conventional measures of binary IoT malware detection, which encompasses both the predictive and computational efficiency. Let TP, TN, FP, and FNdenote the numbers of true positives, true negatives, false positives, and false negatives, respectively, where the positive class corresponds to Backdoor Malware and the negative class corresponds to Benign.

3.5.1. Accuracy

Accuracy measures the overall proportion of correctly classified instances and calculated as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (8)$$

3.5.2. *Precision*

Precision quantifies the reliability of positive predictions (i.e., the proportion of predicted malware that is truly malware) and it calculated as follows:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (9)$$

3.5.3. *Recall (Detection Rate)*

Recall (also called sensitivity or true positive rate) measures the proportion of actual malware instances that are correctly detected and calculated as follows:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (10)$$

3.5.4. *F1-score*

The F1-score provides a harmonic mean of Precision and Recall, balancing missed detections and false alarms and calculated as follows:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}} \quad (11)$$

3.5.5. Execution Time

Along with predictive measures, execution time was also measured of each model to determine the computational overhead. The time metric is the end-to-end execution time to produce cross-validated predictions on a specific classifier (including both model fitting and inference in the evaluation process). This measure is vital in the context of the deployment of IoTs, where real-time or near-real-time detection might be needed with limited computational resources.

Combined, these measures offer a balanced perspective of detection performance and practical viability and enable objective comparison of the assessed machine learning models, backdoor malware detection in IoT networks.

4. Results and Discussion

This part provides the experimental environment and findings of the binary backdoor malware detection benchmark and a discussion of the comparative performance of the tested machine learning models. The analysis provides an overview of the performance in terms of Accuracy, Precision, Recall, F1-score, and the execution time, indicating the most efficient model families and the trade-off between the detection ability and the cost of computations in practice. Moreover, confusion-matrix-based observations are utilized to understand the patterns of errors as well as to make the conclusions that are relevant to deployments in the context of IoT security.

4.1. Experimental Setting

Each experiment is based on a binary IoT backdoor malware detection task, in which a network-traffic record is either Backdoor Malware (positive class) or Benign (negative class). The experiments were performed on Windows 11 Pro 64-bit (version 21H2) with Intel Core i7-1065G7 CPU (1.30-1.50 GHz) and 16 GB of RAM, Spyder was run within Anaconda (Anaconda Navigator 2.6.6), and Python 3.13. Table 3 summarizes the hyperparameter settings of all the classifiers that were evaluated.

Table 3. Hyperparameter configuration of the evaluated ML classifiers

Model	Hyperparameters
XGBoost	objective=binary:logistic
LightGBM	n_estimators=100; learning_rate=0.1; num_leaves=31; max_depth=-1; subsample=1.0; colsample_bytree=1.0; min_child_samples=20; reg_alpha=0.0; reg_lambda=0.0
CatBoost	verbose=0
RandomForest	n_estimators=100; criterion=gini; min_samples_split=2; min_samples_leaf=1; max_features=sqrt; bootstrap=True
ExtraTrees	n_estimators=100; criterion=gini; min_samples_split=2; min_samples_leaf=1; max_features=sqrt; bootstrap=False
Bagging	n_estimators=10; max_features=1.0; bootstrap=True; bootstrap_features=False; oob_score=False
AdaBoost	n_estimators=50; learning_rate=1.0
GradientBoosting	n_estimators=100; learning_rate=0.1; max_depth=3; subsample=1.0; criterion=friedman_mse
HistGradientBoost	max_iter=100; learning_rate=0.1; max_bins=255; min_samples_leaf=20; l2_regularization=0.0
DecisionTree	criterion=gini; splitter=best; min_samples_split=2; min_samples_leaf=1
LogisticRegression	C=1.0; solver=lbfgs; max_iter=1000; multi_class=auto; random_state=42
SGD	loss=hinge; penalty=l2; alpha=0.0001; l1_ratio=0.15; max_iter=1000; tol=0.001; learning_rate=optimal; eta0=0.01
PassiveAggressive	C=1.0; max_iter=1000; tol=0.001; loss=hinge; average=False
RidgeClassifier	alpha=1.0; fit_intercept=True; solver=auto
GaussianNB	var_smoothing=1e-09
SVM	C=1.0; kernel=rbf; gamma=scale; degree=3; coef0=0.0; probability=False; max_iter=-1
KNN	n_neighbors=5; weights=uniform; algorithm=auto; leaf_size=30; p=2; metric=minkowski
LDA	solver=svd; store_covariance=False; tol=0.0001
QDA	reg_param=0.0; store_covariance=False; tol=0.0001
MLP	hidden_layer_sizes=(256,128,64); activation=relu; solver=adam; alpha=0.0001; learning_rate=adaptive; learning_rate_init=0.001; max_iter=2000; early_stopping=True; n_iter_no_change=100; random_state=42

4.2. Confusion Matrices of the Evaluated Classifiers

A confusion matrix (CM) of each of the 20 tested ML classifiers was obtained to supplement the aggregate metrics and give a picture of the error level of detection behavior. In the binary task (Backdoor Malware vs Benign), each CM displays the numbers of true positives (TP) and true negatives (TN), and false positives (FP) / false negatives (FN) on the main and off-diagonal, respectively, thus depicting the frequencies of false alarms raised by a model (Benign → Backdoor Malware) or false negatives (Backdoor Malware → Benign Since the

benchmark data is balanced, the confusion matrices give a visual representation of high accuracy as a natural separation and not as a class dominance. The CMs of most models have a predominance of diagonal tunings, which implies that the benign and backdoor traffic in the analyzed feature space can be separated largely, with the best ensemble learners having near-perfect diagonal structure, i.e., very low FP and FN rates and in line with their very high Accuracy/F1 values. Figure 2–21. Show Confusion matrices of each ML classifier.

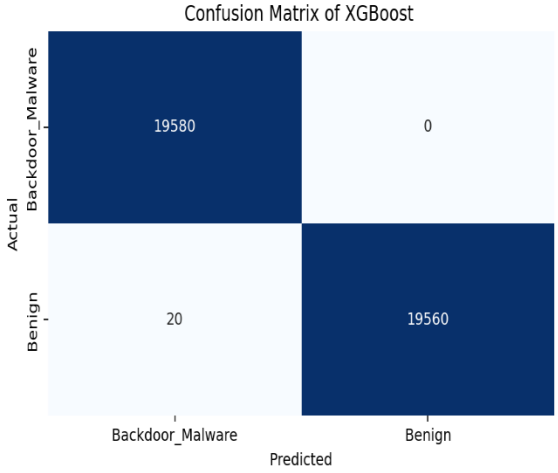

Figure 2. XGBoost

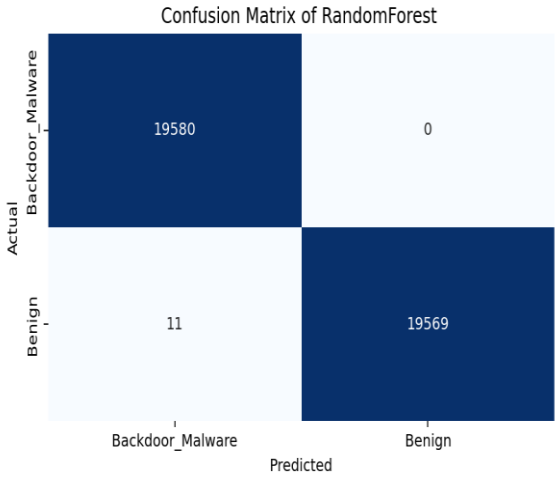

Figure 3. Random Forest

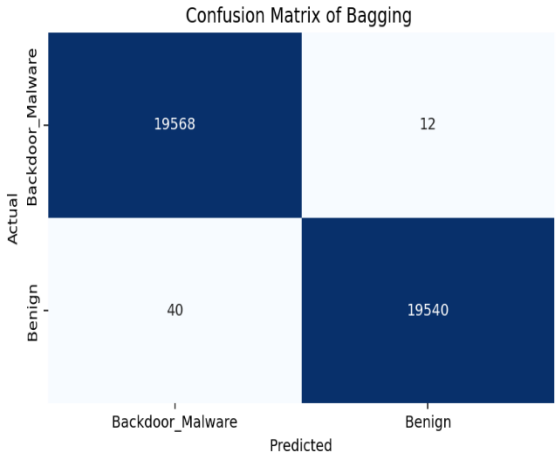

Figure 4. Bagging

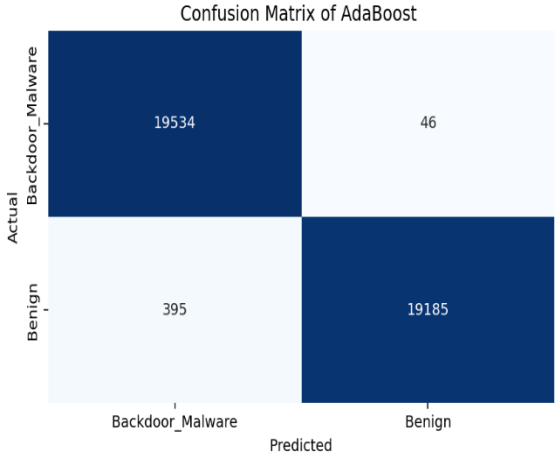

Figure 5. AdaBoost

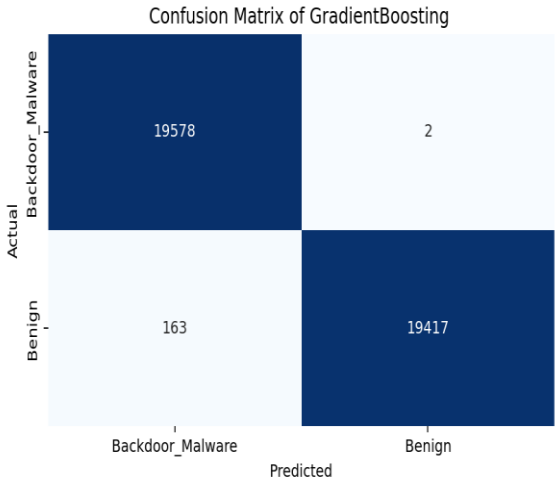

Figure 6. Gradient Boosting

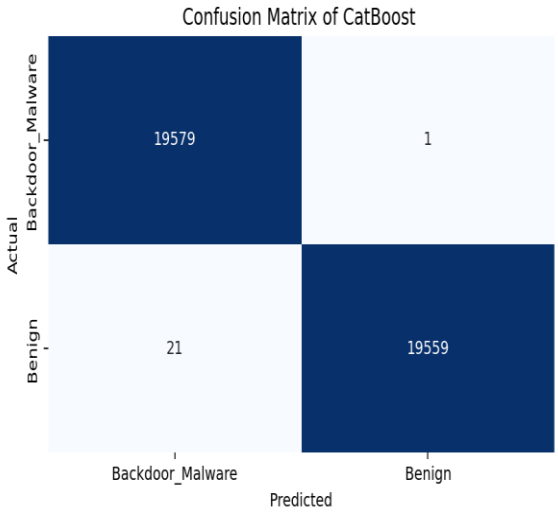

Figure 7. CatBoost

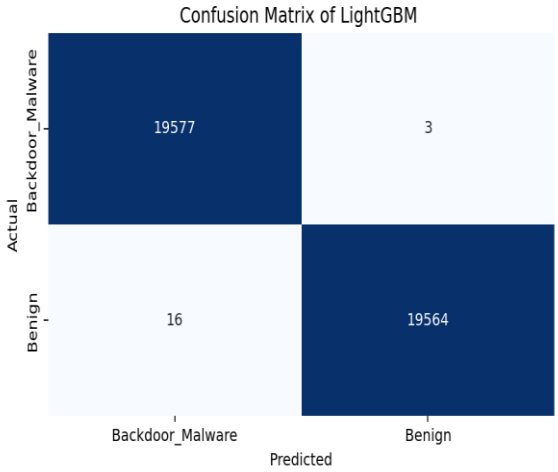

Figure 8. LightGBM

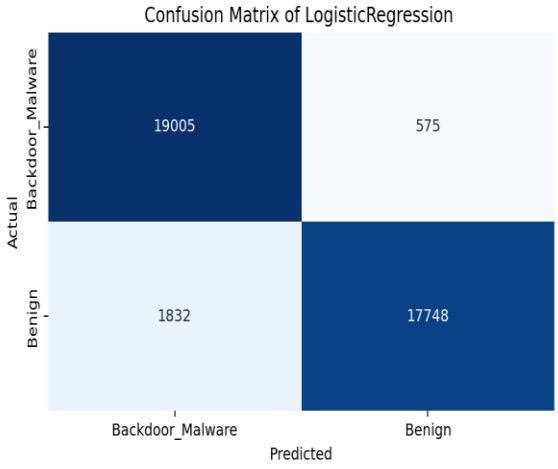

Figure 9. Logistic Regression

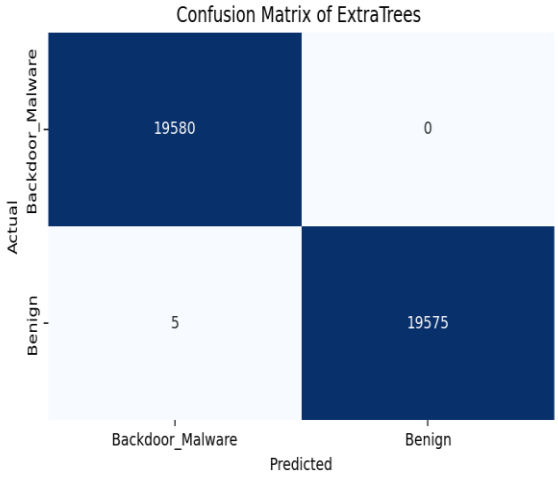

Figure 10. Extra Trees

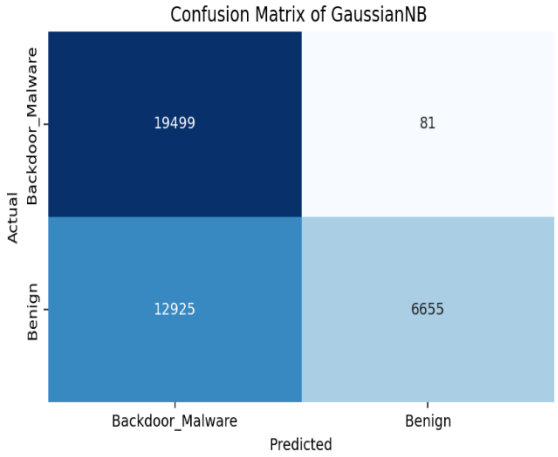

Figure 11. GaussianNB

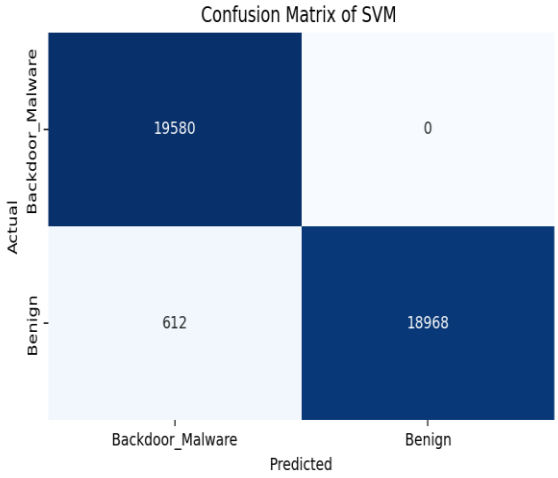

Figure 12. SVM

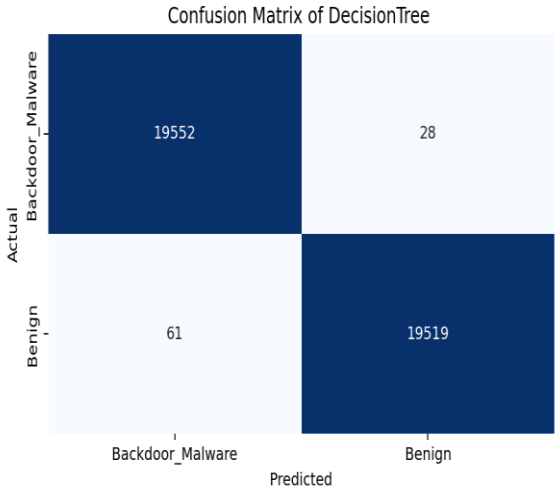

Figure 13. Decision Tree

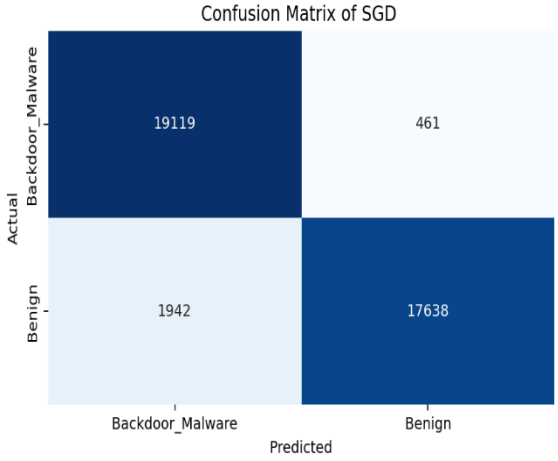

Figure 14. SGD

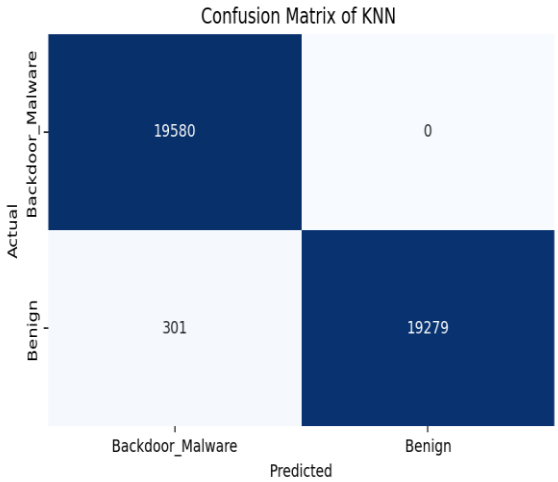

Figure 15. KNN

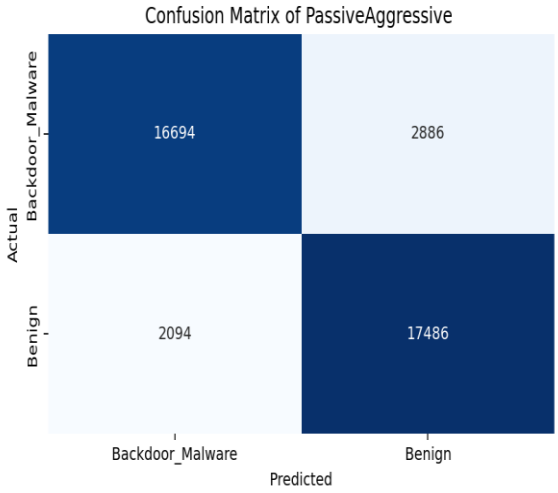

Figure 16. Passive-Aggressive

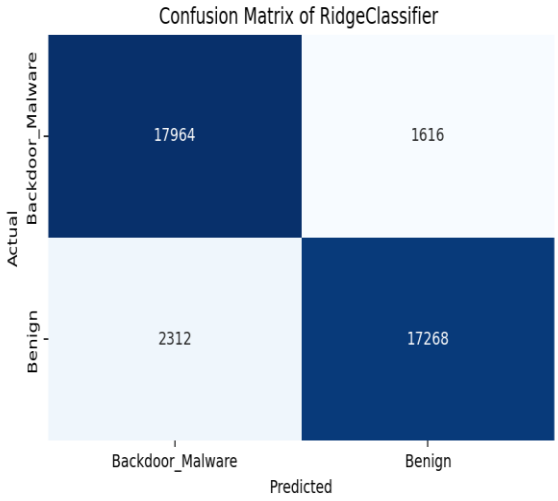

Figure 17. RidgeClassifier

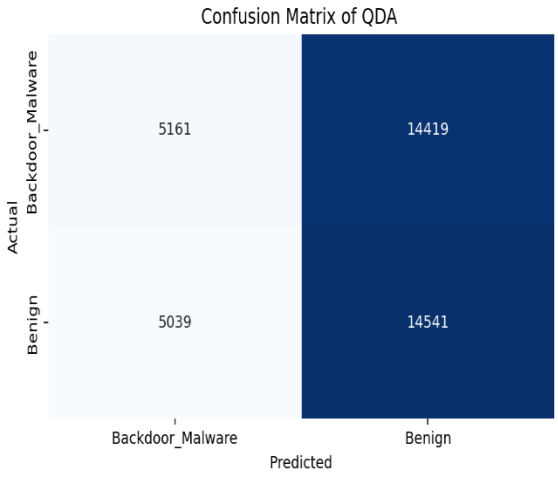

Figure 18. QDA

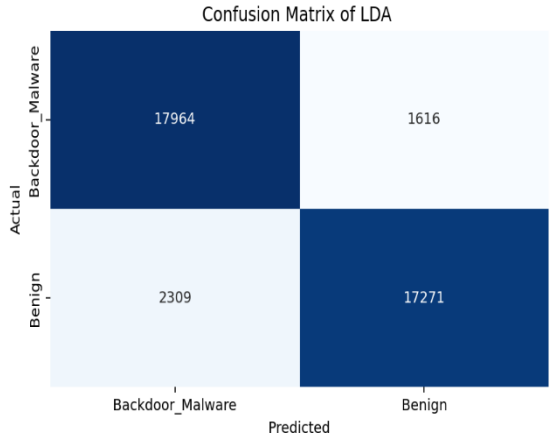

Figure 19. LDA

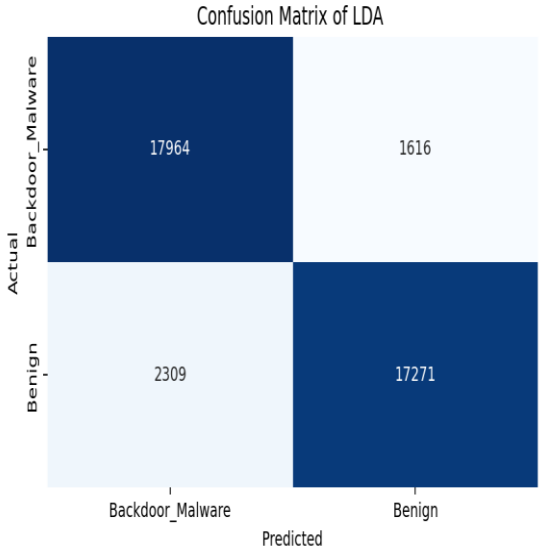

Figure 20. HistGradientBoost

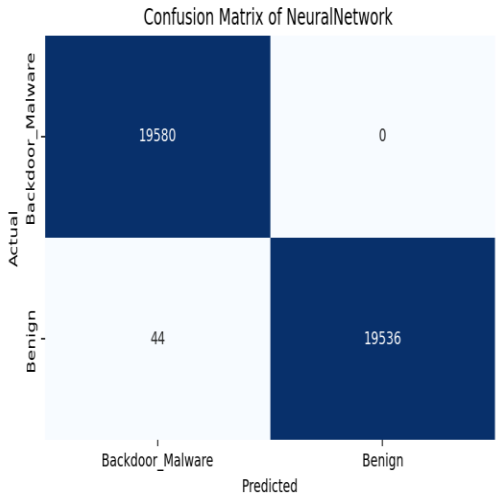

Figure 21. Neural network

4.3. Classification Results

This section presents the quantitative classification findings of the 10-fold cross-validated predictions of all ML classifiers under consideration. Since the task is binary classification and the benchmark dataset is balanced, the reported Accuracy, Precision, Recall, F1-score and execution time give a clear picture of the detection effectiveness and cost. Generally, the findings indicate a high level of dominance of tree-based ensemble learners, which have almost perfect discrimination, as compared to simple linear/discriminate baselines that have significantly poorer detection.

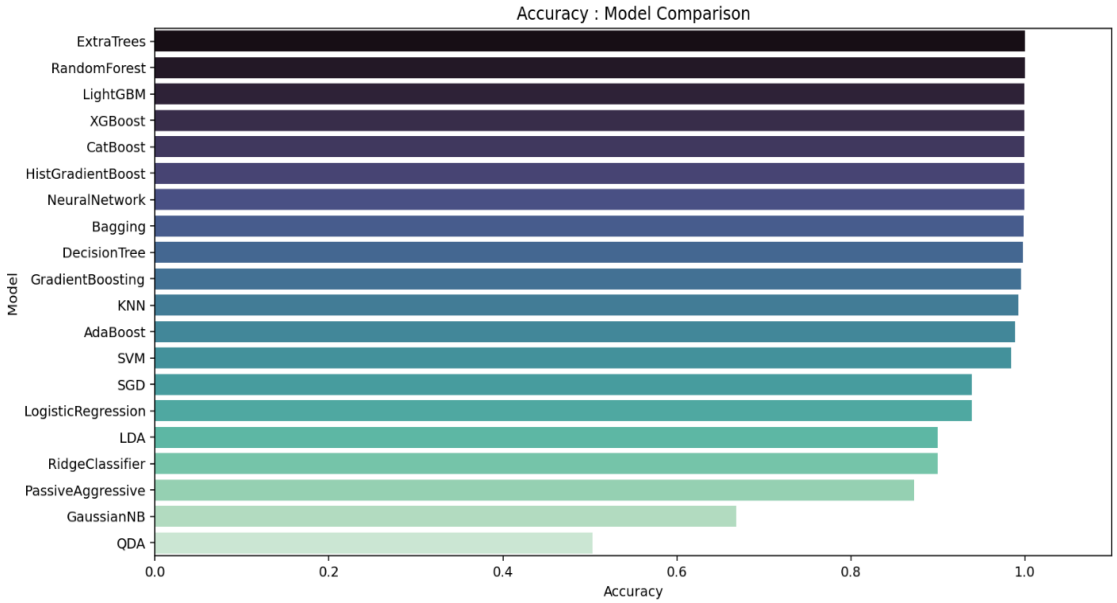

Figure 22. ML classifies Accuracy

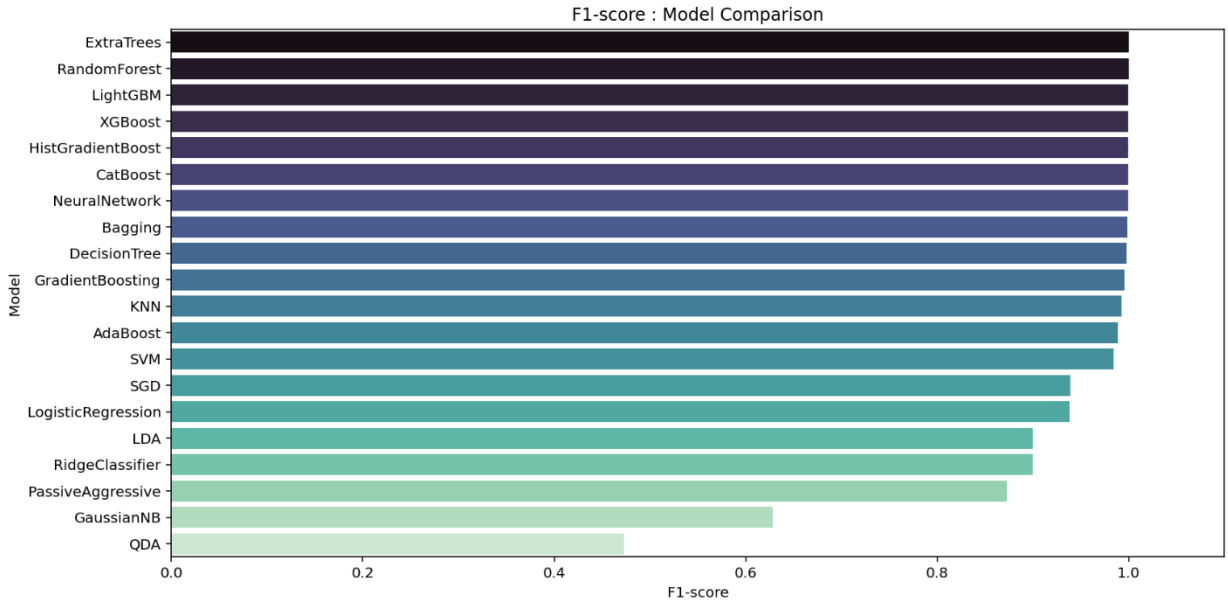

Figure 23. ML classifies F1-score

According to Fig. 22 and Fig. 23 tree-based ensemble learners dominate the benchmark, which means that the extracted traffic features have strong nonlinear patterns and interaction that can be well represented by the ensemble decision boundaries. ExtraTrees is the best-performing classifier with Accuracy = 0.999872 and F1-score = 0.999872 , then there is the RandomForest (Accuracy/F1 = 0.999719). Near-perfect detection accuracy

was also obtained with modern gradient-boosted decision tree models, with LightGBM (Accuracy/F1 = 0.999515) and XGBoost (Accuracy/F1 = 0.999489) being close behind the best models. Equally good performance was achieved by HistGradientBoost and CatBoost (Accuracy/F1 = 0.999438) which validates the success of boosting-based tree learners in detecting tabular malware on IoT.

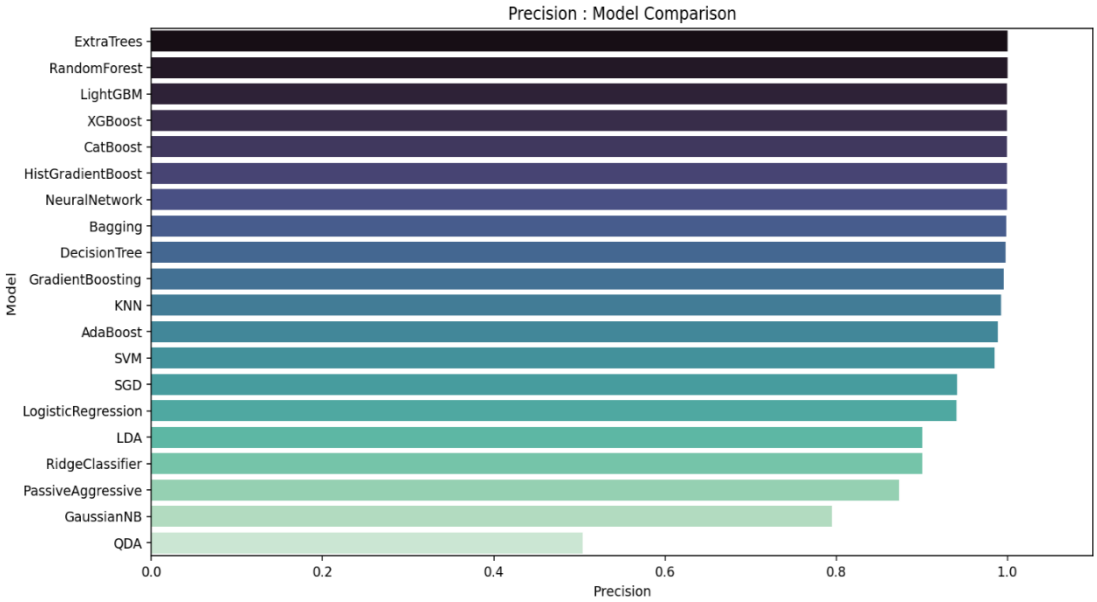

Figure 24. ML classifies Precision

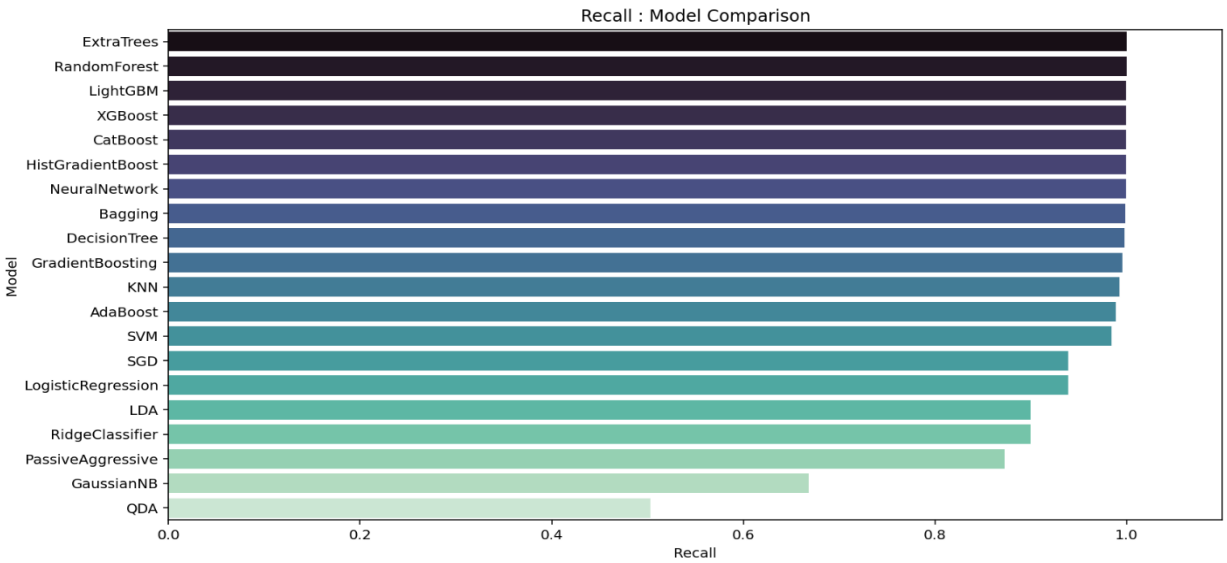

Figure 25. ML classifies Recall

The patterns are consistent in Fig. 24 (Precision) and Fig. 5 (Recall) where the best ensemble techniques are at the same time returning to very high precision (false alarms are minimized) and high recall (malware is missing). This tradeoff is essential in the IoT security context as either of these two error categories can have operational effects: False positives can saturate incident response pipelines, whereas false negatives can facilitate continued compromise and lateral movement. The fact that the rankings are consistent between Accuracy, Precision, Recall, and F1-score also suggests that the best models are not maximizing one of the measures at the cost of another; instead, they are offering consistent high discrimination between benign and backdoor traffic. A second level of models contain Bagging (F1 = 0.998672) and DecisionTree (F1 = 0.997727) models, models that achieve high detection, but do not perform as well as the best ensembles. KNN (F1 = 0.992313) and SVM (F1 = 0.984368) are also competitive baselines but their difference with boosted/tree

ensembles is evident in Fig. 235. However, linear and discriminant baselines (including Logistic Regression (F1 = 0.938471), SGD (F1 = 0.938548), LDA (F1 = 0.899739), and RidgeClassifier (F1 = 0.899662)) demonstrate lower performance, indicating that a purely linear separation cannot be used to capture the underlying GaussianNB (F1 = 0.627840) and QDA (F1 = 0.472872) had the lowest results, which suggests that their distribution assumptions do not fit the feature distribution of this data well.Table 6. Classification performance of each model that is cross validated.

In addition to predictive performance, computational overhead is also vital to the applications of the IoT, especially when detection needs to work in near-real-time or on a constrained platform. Fig. 6 (Time) indicates that there is a significant difference in runtime between models. A few of the best performing methods are also relatively fast, with the most promising being XGBoost (4.090 s) and LightGBM (6.751 s), which are appealing options as deployment-friendly backdoor detectors. An effective balance between near-perfect accuracy and moderate runtime is also available with HistGradientBoost (10.939 s) and ExtraTrees (16.884 s). On the other hand, certain models are very expensive even when they are highly accurate, such as CatBoost (123.192 s), and computational loads are even more severe with the shallow neural model MLP (1491.217 s), which is many times slower than boosted tree structures but herewith offers marginally worse performance than the best ensembles. These findings highlight the fact that when predictive performance techniques are saturated, runtime is a critical consideration in choosing models to use in practical IoT security pipelines.

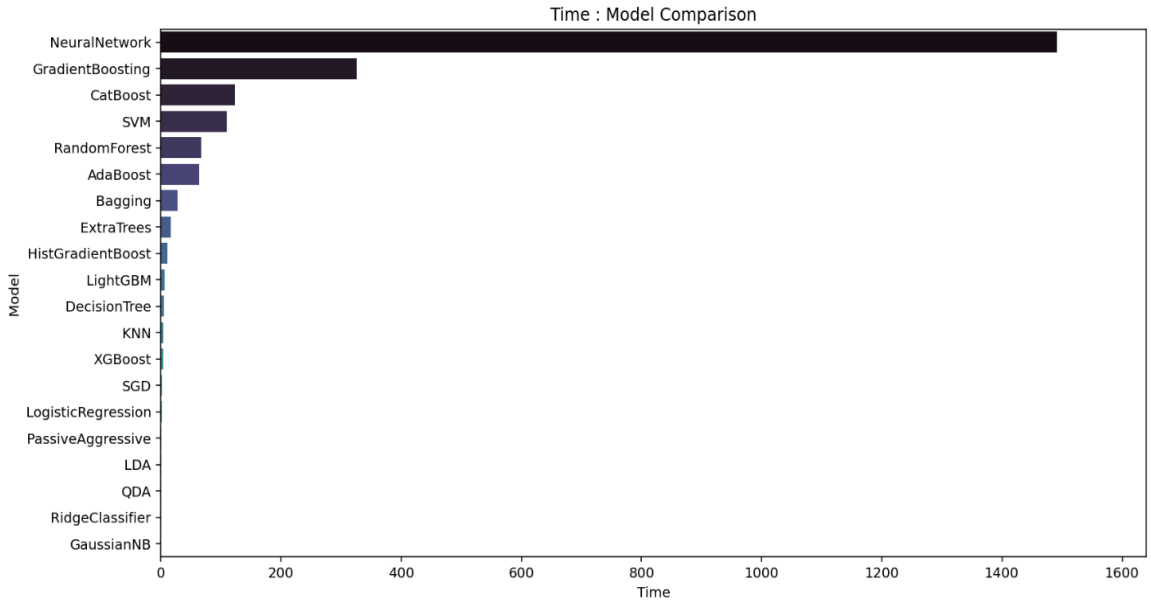

Figure 26. ML classifies Execution Time

The findings suggest that tree-based ensembles and gradient-boosted tree models can offer the most robust and applicable performance in binary IoT backdoor malware detection with near-perfect discrimination at recommended computational efficiency (particularly, in XGBoost and LightGBM).

For tabular IoT traffic features, the superiority of ExtraTrees, RandomForest, LightGBM & XGBoost is expected, since these boosters do not require any manual feature engineering: (i) they can capture the non-linear relationships between features and the class, (ii) handle feature distribution heterogeneity and outliers very well, and (iii) reduce variance by ensembling, thus enhancing stability and generalization. Linear models (such as Logistic Regression, SGD, Ridge) on the contrary enforce a near-linear decision boundary and may underfit when class separation is based on the nonlinear combinations of features, which are well possible in network-traffic representations. Poor performance of these methods is due to the assumption of conditional independence of traffic features in the first case and Gaussian class densities in the latter. While both KNN and SVM can model nonlinear boundaries, they are less effective with distance measures, kernel settings, and scaling, plus they might be less effective in the high-dimensional space than tree ensembles when it comes to

complex interactions among features. If computational efficiency is the primary constraint, LightGBM/XGBoost are recommended due to strong accuracy and low runtime; if maximal accuracy is prioritized and moderate runtime is acceptable, ExtraTrees provides the best overall detection performance in this benchmark.

5. Conclusion and Future Work

This paper has provided a comparative benchmark of classical machine learning classifiers on binary backdoor malware detection on IoT networks whereby instances of traffic are either Backdoor Malware or Benign and the results were consistently provided that the tree-based ensemble learning has the highest detection performance given the conditions evaluated. ExtraTrees had the best overall performance (Accuracy/F1 ≈ 0.999872), then Random Forest (≈ 0.999719), and modern gradient-boosted decision tree frameworks in particular, LightGBM and XGBoost, had near-perfect discrimination with good computational efficiency and suggest that ensemble methods are more effective at nonlinear interactions of features in IoT traffic than linear Future directions can enhance practical usability by confirming generalization to new datasets and to new IoT scenarios, by investigating performance under more natural imbalanced and changing traffic streams, by exploring robustness to concept drift and adversarial evasion, and by enhancing deployment readiness, such as by latency/memory-profiling and lightweight inference optimization to ensure edge or gateway operation; all these directions can help to make ML-driven backdoor.

References

1. M. Sayed, "The internet of things (iot), applications and challenges: a comprehensive review," *Journal of innovative intelligent computing and emerging technologies (JIICET),* vol. 1, no. 01, pp. 20-27, 2024.
2. Y. A. Maz, M. Anbar, S. Manickam, and M. M. Abualhaj, "Utilizing Deep Learning, Ensemble Learning, and Transfer Learning for Enhancing Security in Internet of Things Networks through Intrusion Detection Systems," in *2025 12th International Conference on Information Technology (ICIT)*, 2025: IEEE, pp. 73-76.
3. O. O. Olaniyi, O. J. Okunleye, S. O. Olabanji, C. U. Asonze, and S. A. Ajayi, "IoT security in the era of ubiquitous computing: A multidisciplinary approach to addressing vulnerabilities and promoting resilience," *Asian Journal of Research in Computer Science,* vol. 16, no. 4, pp. 354-371, 2023.
4. S. S. Mahadik, P. M. Pawar, and R. Muthalagu, "Heterogeneous IoT (HetIoT) security: techniques, challenges and open issues," *Multimedia Tools and Applications,* vol. 83, no. 12, pp. 35371-35412, 2024.
5. M. Abualhaj *et al.*, "Comparative Analysis of LSTM-Based Variant Models for Detecting Attacks in IoT Networks," *Journal of Computing & Biomedical Informatics,* vol. 10, no. 01, 2025.
6. N. Patel and A. Singh, "Security issues, attacks and countermeasures in layered IoT ecosystem," *Int. J. Next-Gener. Comput,* vol. 14, p. 400, 2023.
7. A. A. Abu-Shareha, M. M. Abualhaj, A. Hussein, O. Almomani, A. Amer, and A. Achuthan, "Supervised machine learning intrusion detection review and multi-criteria evaluation," *Scientific Reports,* 2026.
8. M. Aqeel, F. Ali, M. W. Iqbal, T. A. Rana, M. Arif, and M. R. Auwul, "A review of security and privacy concerns in the internet of things (IoT)," *Journal of sensors,* vol. 2022, no. 1, p. 5724168, 2022.
9. O. Almomani *et al.*, "A robust model for android malware detection via ML and DL classifiers," *Mesopotamian Journal of Big Data,* vol. 2025, pp. 261-277, 2025.
10. J. Tapiador, "Malware," in *The Computer Security Workbook: A Course Companion Resource*: Springer, 2025, pp. 139-166.
11. Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions," *Electronics,* vol. 12, no. 6, p. 1333, 2023.
12. O. Almomani, A. Alsaaidah, M. A. Almaiah, A. Alzaqebah, M. M. Abualhaj, and W. J. Alzyadat, "Evaluating Machine Learning Classifiers for Detecting Distributed Denial of Service Attacks," in *2025 12th International Conference on Information Technology (ICIT)*, 2025: IEEE, pp. 134-140.
13. M. Almaiah, L. Saqr, L. Al-Rawwash, L. Altellawi, R. Al-Ali, and O. Almomani, "Classification of cybersecurity threats, vulnerabilities and countermeasures in database systems," *Computers, Materials, & Continua,* vol. 81, no. 2, p. 3189, 2024.
14. P. Victor, A. H. Lashkari, R. Lu, T. Sasi, P. Xiong, and S. Iqbal, "IoT malware: An attribute-based taxonomy, detection mechanisms and challenges," *Peer-to-peer Networking and Applications,* vol. 16, no. 3, pp. 1380-1431, 2023.
15. O. Almomani, A. M. Arabiat, A.-A. Hind, and E. Alsariera, "Hybridization of Deep Learning Models for Multiclass Attack Detection in Wireless Sensor Networks," *Journal of Communications,* vol. 21, no. 1, 2026.
16. M. N. Ghouri, G. Ahmed, A. S. Al-Shamayleh, M. Maaz, S. Siddiqui, and A. Akhunzada, "Next-Gen IoT Security: Deep Learning-Based Detection of RPL Attacks in Mobile Converged Networks," *IEEE Open Journal of the Communications Society,* 2026.
17. M. Abdelhaq, A. S. Al-Shamayleh, A. Akhunzada, N. Ivković, and T. Hasan, "Scalable and Resilient AI Framework for Malware Detection in Software-Defined Internet of Things," *Computers, Materials, & Continua,* vol. 87, no. 1, 2026.
18. T. M. Ghazal, A. A. Ahmed, K. Madinabonu, A. Ibrahim, F. Salomova, and J. Kavitha, "Federated Learning with Secure Aggregation for Distributed Malware Detection in IoT Networks," in *2025 International Conference on Electrical Engineering and Informatics (ICEEI)*, 2025: IEEE, pp. 1-7.
19. A. Arabiat and M. Altayeb, "Enhancing internet of things security: evaluating machine learning classifiers for attack prediction," *International Journal of Electrical & Computer Engineering (2088-8708),* vol. 14, no. 5, 2024.
20. K. Paraskevas and T. Christos, "Data preprocessing and feature engineering for data mining: Techniques, tools, and best practices," *AI,* vol. 6, no. 10, p. 257, 2025.
21. A. R. Khaliq, S. Ullah, T. Ahmad, A. Yadav, and M. I. Majid, "Behavioral Analysis of Backdoor Malware Exploiting Heap Overflow Vulnerabilities Using Data Mining and Machine Learning," *Pakistan Journal of Engineering, Technology and Science,* vol. 11, no. 1, pp. 1-13, 2023.

22. M. M. Khan, A. Buriro, T. Ahmad, and S. Ullah, "Backdoor malware detection in industrial IoT using machine learning," *Computers, Materials & Continua*, vol. 81, no. 3, pp. 4691-4705, 2024.
23. C. S. Dash, "LightGBM-Powered Solutions for Backdoor Malware Detection in SCADA Networks," in *2024 International Conference on Communication, Computing and Energy Efficient Technologies (I3CEET)*, 2024: IEEE, pp. 1765-1771.
24. D. Panda, N. Padhy, and K. Sharma, "Strengthening IoT Resilience: A Study on Backdoor Malware and DNS Spoofing Detection Methods," in *2025 International Conference on Emerging Systems and Intelligent Computing (ESIC)*, 2025: IEEE, pp. 795-800.
25. E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.
26. O. Almomani *et al.*, "A metaheuristic feature selection model using bat optimization for malicious URL attack detection," *Scientific Reports*, 2026, doi: https://doi.org/10.1038/s41598-026-51981-2.
27. O. Almomani, A. Almomani, and S. Memon, "Enhancing IoT Security Using Machine Learning for Port Scan Attack Detection," *Procedia Computer Science*, vol. 275, pp. 532-540, 2026.
28. Q. Shambour, M. Al-Zyoud, and O. Almomani, "Quantum-inspired hybrid metaheuristic feature selection with SHAP for optimized and explainable spam detection," *Symmetry*, vol. 17, no. 10, p. 1716, 2025.
29. A. Almomani *et al.*, "Ensemble-based approach for efficient intrusion detection in network traffic," *Intelligent Automation & Soft Computing*, vol. 37, no. 2, 2023.
30. O. Almomani, A. Alsaaidah, A. A. A. Shareha, A. Alzaqebah, and M. Almomani, "Performance Evaluation of Machine Learning Classifiers for Predicting Denial-of-Service Attack in Internet of Things," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 1, 2024.
31. A. Alsaaidah, O. Almomani, A. A. Abu-Shareha, M. M. Abualhaj, and A. Achuthan, "ARP spoofing attack detection model in IoT network using machine learning: Complexity vs. accuracy," *Journal of Applied Data Sciences*, vol. 5, no. 4, pp. 1850-1860, 2024.
32. A. Abu-Khadrah, M. A. AlMutairi, M. R. Hassan, and A. M. Ali, "Enhancing IoT Security and Malware Detection Based on Machine Learning," in *International Congress on Information and Communication Technology*, 2025: Springer, pp. 561-571.
33. M. Madi, F. Jarghon, Y. Fazea, O. Almomani, and A. Saaidah, "Comparative analysis of classification techniques for network fault management," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, no. 3, pp. 1442-1457, 2020.