Evaluating Machine Learning Models for Network Anomaly Detection in IoT Environments

Bilal ABDULHADI
Faculty of Engineering
Biruni University
Istanbul, Turkey
251132903@st.biruni.edu.tr

Dr. MHD RAJA ABOU HARB
Faculty of Engineering
Biruni University
Istanbul, Turkey
mharb@biruni.edu.tr

***Abstract*—The growth of Internet of Things (IoT) networks has increased the need for effective intrusion and anomaly detection methods. Many IoT devices have limited resources and can be vulnerable to attacks such as DDoS, ransomware, password attacks, port scanning, and injection-based attacks. This paper evaluates traditional machine learning models for binary network anomaly detection using the ML-EdgeIIoT dataset. A cleaned preprocessing pipeline is used to remove duplicate records, high-cardinality text-based features, possible leakage or proxy columns, and constant or near-constant features. Five models are evaluated: Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, and XGBoost. The models are compared using accuracy, precision, recall, F1-score, false positive rate, and prediction time. The results show that XGBoost achieved the best overall performance with an accuracy of 0.9753, recall of 0.9991, and F1-score of 0.9856. Logistic Regression achieved the lowest false positive rate, but its low recall means that it missed many attack samples. Overall, XGBoost gave the best balance for detecting attack traffic in the cleaned IoT network dataset.**

***Index Terms*—IoT security, anomaly detection, intrusion detection, machine learning, Edge-IIoTset, XGBoost**

I. Introduction

The Internet of Things (IoT) is now used in many areas such as smart homes, healthcare, transportation, agriculture, and industrial automation. IoT devices collect and exchange data through network connections, which makes them useful for monitoring and automation. However, the fast growth of IoT networks also creates important security problems. Many IoT devices have limited memory, processing power, and energy, so they cannot always run strong security mechanisms. Because of this, IoT networks can be exposed to attacks such as Distributed Denial of Service (DDoS), Man-in-the-Middle (MITM), password attacks, ransomware, port scanning, and injection attacks [1], [2].

Traditional intrusion detection systems can detect known attacks using predefined rules or signatures. However, these systems are not always effective when the attack is new or when the attack behavior changes. For this reason, machine learning-based intrusion detection has become an important direction in IoT security. Machine learning models can learn patterns from network traffic and classify traffic as normal or malicious. Previous studies have used models such as Decision Tree, Random Forest, K-Nearest Neighbors, XGBoost, and deep learning models for IoT intrusion detection [3]–[5].

Although many previous studies achieved strong results, their results are not always easy to compare directly. This is because they use different datasets, preprocessing methods, feature sets, classification tasks, and evaluation metrics. Some studies focus mainly on accuracy, but in intrusion detection, other metrics are also important. For example, low recall means that some attacks are missed, while a high false positive rate means that normal traffic is wrongly detected as attack traffic. Therefore, it is useful to evaluate several models under the same preprocessing and testing conditions.

This paper evaluates traditional machine learning models for binary network anomaly detection in IoT environments using the ML-EdgeIIoT dataset, which is part of the Edge-IIoTset cybersecurity dataset [1]. The classification task uses the `Attack_label` column, where class 0 represents normal traffic and class 1 represents attack traffic. Five models are tested in this work: Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), and XGBoost.

The main points of this work are:

- A cleaned preprocessing pipeline is applied to the ML-EdgeIIoT dataset before model training.
- Five traditional machine learning models are trained and tested under the same conditions.
- The models are compared using accuracy, precision, recall, F1-score, false positive rate, and prediction time.
- The best model is selected based on the balance between detection performance and practical behavior.

The rest of this paper is organized as follows. Section II presents the related work. Section III explains the proposed methodology, including the dataset, preprocessing, models, and evaluation metrics. Section IV presents the experimental results and discussion. Section V gives the conclusion and future work.

II. Related Work

Machine learning has been used in many recent studies for IoT intrusion and anomaly detection. The main idea in these studies is to train models on network traffic data and classify the traffic as normal or malicious. Different researchers used different datasets, preprocessing steps, and models, so their results are not always directly comparable.

Lee and Majd proposed machine learning-based frameworks for anomaly detection and attack classification in IoT networks using the Kitsune dataset [3]. They tested several classical machine learning models, including Logistic Regression, Decision Tree, Random Forest, Naive Bayes, Support Vector Machine, KNN, Gradient Boosting, XGBoost, and Extra Trees. They also used ANOVA and Chi-square feature selection methods. Their results showed that Random Forest and XGBoost performed well for binary anomaly detection, while Random Forest gave the best result for family-based attack classification.

Zhang proposed a supervised machine learning-based intrusion detection system for securing IoT networks using the BoTNeTIoT-L01 dataset [4]. The study compared Decision Tree, Random Forest, and KNN for binary intrusion detection. The results showed very high performance for Decision Tree and Random Forest. This supports the idea that tree-based models can be effective for IoT intrusion detection.

Ramdan et al. proposed a hybrid CNN-RNN model for IoT network intrusion detection using the CIC IoT 2022 dataset [5]. Their model combined CNN layers for spatial feature extraction and RNN/LSTM layers for temporal behavior learning. The hybrid model achieved strong performance and outperformed standalone CNN and RNN models. However, deep learning models may require more computational resources, which can be a limitation for resource-constrained IoT environments.

Kour et al. studied self-supervised learning for anomaly detection in IoT networks using the TON_IoT dataset [6]. Their approach used contrastive learning, autoencoders, and transformers. This direction is useful because it can reduce the need for labeled data. However, it also makes the model more complex and may require more careful training and tuning.

Srivastava et al. reviewed different machine learning algorithms for IoT attack detection, including Random Forest, SVM, KNN, Naive Bayes, Artificial Neural Networks, K-Means, Decision Trees, Logistic Regression, RNN, LSTM, Autoencoders, and XGBoost [2]. The review explained that there is no single best model for all IoT attack detection cases, because the best model depends on the dataset, attack type, computational cost, and system requirements.

From these studies, it can be seen that machine learning and deep learning methods are useful for IoT intrusion detection. However, many studies use different datasets and different experimental settings. Because of this, it is useful to compare several models under the same dataset, same preprocessing pipeline, and same evaluation metrics. In this paper, five traditional machine learning models are evaluated on the cleaned ML-EdgeIIoT dataset for binary IoT network anomaly detection.

III. Proposed Methodology

This section explains the methodology used in this paper. The goal is to evaluate traditional machine learning models for binary IoT network anomaly detection. The full process includes dataset checking, data cleaning, preprocessing, model training, and evaluation.

A. System Architecture

The proposed system is an experimental machine learning pipeline for binary IoT anomaly detection. It does not introduce a new IoT communication architecture. Instead, it shows the main steps used to prepare the dataset and compare the selected machine learning models fairly.

The pipeline starts by loading and checking the ML-EdgeIIoT dataset. In this stage, the dataset shape, missing values, duplicated rows, feature types, and class distribution are examined. After that, the data cleaning stage removes duplicate rows and excludes selected columns that may reduce generalization, such as high-cardinality text-heavy columns, possible leakage or proxy columns, and constant or near-constant columns.

After cleaning, the remaining numeric features are prepared using median imputation and standardization. The dataset is then divided into training and testing subsets using stratified sampling. Finally, the selected machine learning models are trained and evaluated using the same test set. The overall workflow of the proposed system is shown in Fig. 1.

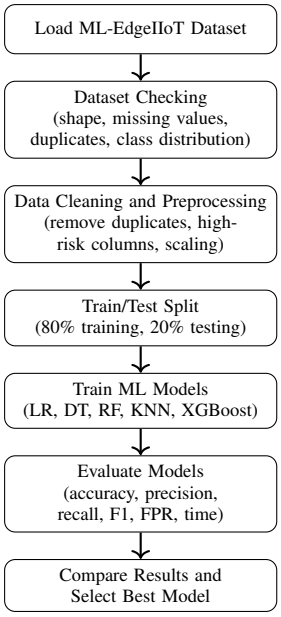

Fig. 1. System architecture of the proposed machine learning pipeline for IoT anomaly detection.

B. Dataset Description

This study uses the ML-EdgeIIoT dataset, which is part of the Edge-IIoTset cybersecurity dataset for IoT and Industrial IoT applications [1]. The dataset is suitable for intrusion detection experiments because it contains both normal traffic and different types of attack traffic.

The original dataset used in this experiment contains 157,800 records and 63 columns. The binary classification target is the `Attack_label` column. In this column, class 0

represents normal traffic, and class 1 represents attack traffic. Before duplicate removal, the dataset contains 24,301 normal samples and 133,499 attack samples. The `Attack_type` column is not used as an input feature because it contains the attack category and may introduce label-related information into the binary classification task.

C. Data Preprocessing

The preprocessing step is important to make the experiment more reliable. First, exact duplicate rows are removed from the dataset. A total of 814 duplicate rows are removed, reducing the dataset from 157,800 records to 156,986 records.

Several groups of columns are removed before training. High-cardinality and text-heavy columns are removed, such as timestamps, IP addresses, URI fields, payload fields, port fields, and message content fields. These columns can make the models learn dataset-specific patterns instead of general traffic behavior. Possible leakage or proxy columns are also removed because they may have a near-direct relationship with the target label. Constant and near-constant columns are also removed because they do not add useful information for classification.

After cleaning, the final feature set contains 27 numeric features. No categorical features remain after this step. The numeric features are processed using median imputation and standardization. The dataset is then divided into training and testing sets using an 80:20 stratified split. Stratification is used to keep the class distribution similar in both sets. The final training set contains 125,588 samples, and the testing set contains 31,398 samples.

D. Machine Learning Models

Five traditional machine learning models are evaluated in this work: Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), and XGBoost. These models were selected because they are commonly used in IoT intrusion detection studies and represent different types of machine learning approaches [2]–[4].

Logistic Regression is used as a simple baseline model. Decision Tree is used as an interpretable tree-based classifier. Random Forest is used as an ensemble model that combines several decision trees. KNN is used as a distance-based classifier. XGBoost is used as a boosting-based tree ensemble model. All models are trained using the same cleaned training data and evaluated using the same testing data.

E. Model Hyperparameters

Table I shows the main hyperparameter settings used for the evaluated models. Default parameters were kept for the remaining settings unless stated otherwise.

TABLE I
MAIN HYPERPARAMETER SETTINGS OF EVALUATED MODELS

Model	Main Hyperparameters
Logistic Regression	`max_iter=1000, class_weight=balanced`
Decision Tree	`class_weight=balanced, random_state=42`
Random Forest	`n_estimators=100, class_weight=balanced, random_state=42, n_jobs=-1`
KNN	`n_neighbors=5`
XGBoost	`eval_metric=logloss, random_state=42, n_jobs=-1`

F. Evaluation Metrics

The models are evaluated using accuracy, precision, recall, F1-score, false positive rate, and prediction time. Accuracy shows the overall percentage of correct predictions. Precision shows how many predicted attack samples are actually attacks. Recall shows how many real attack samples are detected correctly. F1-score gives a balance between precision and recall.

False positive rate is also important in intrusion detection because it shows how many normal samples are wrongly classified as attacks. Prediction time is used to compare how fast each model makes predictions on the test set.

The evaluation metrics are calculated using the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively.

IV. Experimental Results and Discussion

This section presents the results of the evaluated machine learning models on the cleaned ML-EdgeIIoT dataset. The models are compared using accuracy, precision, recall, F1-score, false positive rate, and prediction time.

A. Experimental Setup

The experiment was performed as a binary classification task using the `Attack_label` column. Class 0 represents normal traffic, and class 1 represents attack traffic. After preprocessing, the final dataset contained 27 numeric features. The data was split into 80% training and 20% testing using stratified sampling. The training set contained 125,588 samples, and the testing set contained 31,398 samples.

Five models were trained and evaluated: Logistic Regression, Decision Tree, Random Forest, KNN, and XGBoost. All models were tested on the same cleaned test set to make the comparison fair.

B. Model Performance Comparison

Table II shows the final performance results of all evaluated models.

TABLE II
PERFORMANCE COMPARISON OF MACHINE LEARNING MODELS

Model	Acc.	Prec.	Recall	F1	FPR	Pred. Time (s)
Logistic Regression	0.7171	0.9781	0.6806	0.8027	0.0831	0.0012
Decision Tree	0.9622	0.9768	0.9786	0.9777	0.1272	0.0023
Random Forest	0.9667	0.9747	0.9862	0.9804	0.1395	0.0399
KNN	0.8977	0.9185	0.9646	0.9410	0.4675	2.1052
XGBoost	0.9753	0.9724	0.9991	0.9856	0.1547	0.0058

From Table II, XGBoost achieved the best overall performance. It obtained the highest accuracy of 0.9753, the highest recall of 0.9991, and the highest F1-score of 0.9856. This means that XGBoost was the strongest model for detecting attack traffic in this experiment.

Random Forest also performed well, with an accuracy of 0.9667 and an F1-score of 0.9804. Decision Tree achieved close results with an accuracy of 0.9622 and an F1-score of 0.9777. KNN achieved an F1-score of 0.9410, but it had the highest false positive rate and the slowest prediction time. Logistic Regression had the lowest false positive rate, but its recall was only 0.6806, which means it missed many attack samples.

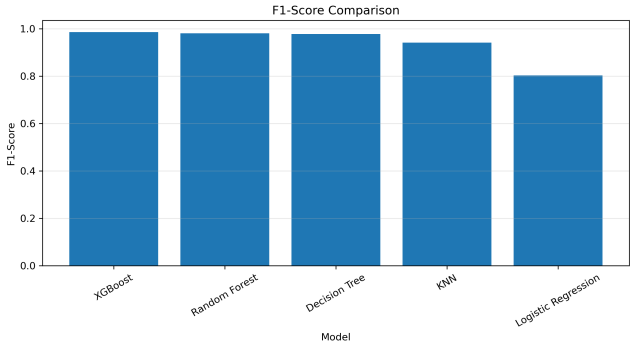

Fig. 2. F1-score comparison of evaluated models.

Fig. 2 shows that XGBoost achieved the highest F1-score, followed by Random Forest and Decision Tree. This confirms that tree-based ensemble models performed better than Logistic Regression and KNN in this experiment.

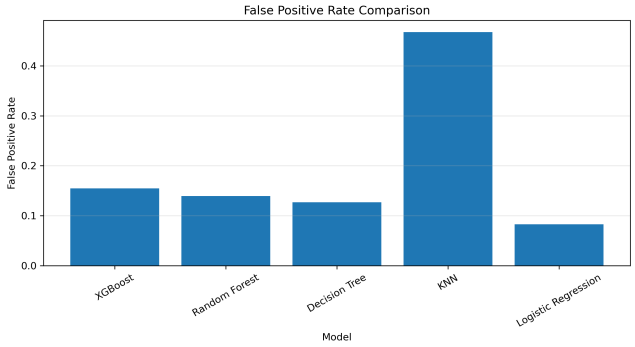

Fig. 3. False positive rate comparison of evaluated models.

Fig. 3 shows the false positive rate for each model. Logistic Regression achieved the lowest false positive rate, but this result should be understood carefully because the same model also had low recall. In security tasks, low recall is dangerous because it means some attacks are not detected.

C. Discussion

The results show that XGBoost is the best overall model in this experiment. Its high recall means that it detected almost all attack samples, and its high F1-score shows a strong balance between precision and recall. This is important in IoT anomaly detection because missing attacks can be more harmful than producing some false alerts.

The high performance values may be related to the structure of the ML-EdgeIIoT dataset and the cleaned feature set used in this experiment. Since the task is binary classification, the models only need to separate normal traffic from attack traffic, not classify each attack type separately. This can make the task easier than multi-class attack classification. Also, removing possible leakage or proxy columns made the evaluation more reliable, but the remaining numerical features still provided enough information for strong detection performance.

Random Forest and Decision Tree also gave strong results. These models are easier to understand than XGBoost and may still be useful when interpretability is important. However, XGBoost gave better overall detection performance, especially in terms of recall and F1-score.

Logistic Regression produced the lowest false positive rate, which means it was better at avoiding wrong attack alerts for normal traffic. However, its recall was much lower than the other models. Because of that, it is not the best choice for attack detection in this dataset. The false positive rate is important because a high value means that normal traffic is incorrectly reported as malicious. In a real IoT security system, this may create too many alerts and reduce trust in the detection system. However, choosing a model only based on false positive rate is not enough; recall and F1-score must also be considered.

KNN was weaker compared with the tree-based models. It had a high false positive rate and slower prediction time. This makes it less suitable for this task, especially if the system needs faster detection.

Although the results are strong, this study has some limitations. First, the experiment is limited to binary classification using the `Attack_label` column, so it does not show how well the models classify specific attack types. Second, the evaluation is based on one train-test split, and cross-validation was not applied. Third, the models were tested on a cleaned offline dataset, so real-time IoT deployment and edge-device constraints were not evaluated. These limitations can be addressed in future work.

Overall, the results suggest that XGBoost is the most effective model for the cleaned ML-EdgeIIoT binary anomaly detection task. However, if the goal is to reduce false alarms only, Logistic Regression may be considered, but with the risk of missing more attacks.

V. Conclusion and Future Work

This paper evaluated five traditional machine learning models for binary network anomaly detection in IoT environments using the cleaned ML-EdgeIIoT dataset. The evaluated models were Logistic Regression, Decision Tree, Random Forest, KNN, and XGBoost. Before training, the dataset was cleaned by removing duplicate rows, high-cardinality text-based columns, possible leakage or proxy columns, and constant or near-constant columns. After preprocessing, 27 numeric features were used for training and testing.

The experimental results showed that XGBoost achieved the best overall performance. It reached an accuracy of 0.9753, recall of 0.9991, and F1-score of 0.9856. This means that XGBoost was the strongest model for detecting attack traffic in this experiment. Random Forest and Decision Tree also achieved strong results, but they were slightly lower than XGBoost. Logistic Regression achieved the lowest false positive rate, but its recall was much lower, which means it missed more attacks. KNN was less suitable because it had a high false positive rate and slower prediction time.

Based on these results, XGBoost can be considered the best model among the tested methods for this cleaned binary IoT anomaly detection task. However, this study is limited to binary classification using the `Attack_label` column. Future work can extend this experiment to multi-class attack classification using the `Attack_type` column. Other future directions include hyperparameter tuning, feature selection methods, deep learning models, and testing the models in a real-time or edge-based IoT environment.

References

[1] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *TechRxiv*, 2022.

[2] A. Srivastava, A. Jindal, A. Shrivastava, and V. Sawan, "A review on iot attack detection using machine learning algorithms," in *2025 6th International Conference on Data Intelligence and Cognitive Informatics (ICDICI)*, 2025, pp. 1098–1103.

[3] K. Lee and N. E. Majd, "Anomaly detection and attack classification in iot networks using machine learning," in *2023 IEEE 42nd International Performance, Computing, and Communications Conference (IPCCC)*, 2023, pp. 453–458.

[4] Y. Zhang, "A machine learning-based intrusion detection system for securing internet of things networks," in *2025 7th International Conference on Information Science, Electrical and Automation Engineering (ISEAE)*, 2025, pp. 374–377.

[5] L. Ramdan, U. Aditiawarman, and J. Asian, "Enhancing iot network intrusion detection with integrated cnn-rnn model," in *2025 IEEE 11th International Conference on Computing, Engineering and Design (ICCED)*, 2025.

[6] H. Kour, P. Bavadiya, and S. Kumar, "Self-supervised learning for anomaly detection in iot networks," in *2025 3rd International Conference on Advancement in Computation and Computer Technologies (InCACCT)*, 2025, pp. 644–648.