Dicle University
Journal of Engineering
https://dergipark.org.tr/tr/pub/dumf
duje.dicle.edu.tr

Research Article

# DoS and DDoS Attacks on Internet of Things and Their Detection by Machine Learning Algorithms

**Mustafa Muhammed ŞİMŞEK[1], Emrah ATILGAN[2*]**

[1] Eskisehir Osmangazi University, Computer Engineering Department, mmustafasimsek23@gmail.com, Orcid No: 0000-0002-2533-4934
[2] Eskisehir Osmangazi University, Computer Engineering Department, emrah.atilgan@ogu.edu.tr.com, Orcid No: 0000-0002-0395-9976

ARTICLE INFO

*Article history*:

Received 17 January 2024
Received in revised form
31 March 2024
Accepted 17 April 2024
Available online 30 June 2024

*Keywords*:

*DoS, DDoS, IoT, Machine Learning, Classification*

*Doi: 10.24012/dumf.1421337*

* Corresponding author

ABSTRACT

Machine Learning (ML) algorithms play a crucial role in fortifying the security of Internet of Things (IoT) environments. In this study, we focus on several key ML algorithms, namely Random Forest, AdaBoost, Decision Trees, Naive Bayes, Logistic Regression, Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN). These algorithms are evaluated within the unique context of IoT security, employing an original dataset meticulously crafted for this study. The dataset OGU-IoT23 is designed to capture the intricacies of cyber threats in an IoT network, featuring attacks such as DDoS, HTTP Flood, SYN Flood, Port Scan, and UDP Flood. This original dataset OGU-IoT23 serves as a foundation for the comprehensive evaluation of ML algorithms, allowing us to assess their effectiveness in identifying and mitigating diverse attack patterns targeting IoT devices. The algorithms are examined based on their performance metrics such as accuracy, F1-score, precision, recall, and test accuracy, emphasizing their suitability for real-world IoT security applications. The results show that Random Forest and AdaBoost are the top performers in terms of performance metrics. The study aims to provide valuable insights into the strengths and limitations of these ML algorithms, aiding researchers and practitioners in developing resilient security measures designed for IoT settings.

## Introduction

Machine Learning (ML) algorithms significantly contribute to improving effectiveness, efficiency and accuracy of various tasks in contemporary computing [1], [2]. This paper explores the performance and applicability of several prominent ML algorithms in the context of an Internet of Things (IoT) application. Specifically, it focuses on algorithms such as Random Forest, AdaBoost, Decision Trees, Naive Bayes, Logistic Regression, Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN). Each algorithm is examined in terms of its characteristics, strengths, and weaknesses, and their performances are evaluated through a comprehensive experimental setup involving attacks on an IoT network.

The research commences by presenting the Random Forest algorithm, an ensemble learning technique extensively employed in tasks related to classification and regression [3]. It discusses its operation, emphasizing its ability to mitigate overfitting through randomness injection during training. Following Random Forest, the paper explores AdaBoost, a robust ensemble learning algorithm designed to enhance weak learners sequentially [4]. Subsequently, Decision Trees are introduced as versatile and interpretable ML algorithms commonly used for classification and regression [5]. The Naive Bayes algorithm, a probabilistic method based on Bayes' theorem, is presented as an efficient tool for classification tasks, especially in natural language processing [6]. Logistic Regression, despite its name, is described as a classification algorithm widely used for binary classification tasks, valued for its simplicity and interpretability [7]. Support Vector Machines (SVMs) are discussed as powerful algorithms for supervised learning, excelling in scenarios with complex decision boundaries [8]. The paper further explores K-Nearest Neighbors (k-NN), a straightforward yet efficient algorithm that depends on proximity for tasks involving classification and regression [9]. Additionally, fundamental performance metrics such as the confusion matrix, accuracy, precision, recall, and F1-score are introduced to assess the effectiveness of the ML algorithms.

In the literature, several datasets have been utilized to test algorithms proposed for intrusion detection. One such dataset is KDD-CUP 99, which is generated from raw network traffic data collected from a simulated military network environment at the University of California. This dataset has been extensively used by various algorithms to classify network connections as "normal" or "intrusion". Faraknakian and Heikkonen [10] tested their Deep Auto-Encoder based Intrusion Detection system on the KDD-CUP99 dataset, achieving a binary classification accuracy of 96%. Khalvati et al. [11] introduced a hybrid learning approach for intrusion detection. They utilized 10% of the KDD-CUP99 dataset and created a new training dataset using K-Medoids clustering. Feature selection was performed using SVM, followed by evaluation with the Naive Bayes classifier.

The CICDDoS2019 dataset was created by a research institute called CIC at the University of New Brunswick in Canada to analyze DDoS attacks and design mitigation techniques. This dataset is considered significant as it covers 12 different types of DDoS attacks and comprises a large volume of 12,794,627 samples. Ullah and Mahmoud [12] introduced another large dataset, IoTID20, containing 625,783 samples from various IoT devices, including normal and four types of attack data: Scan, DoS, Mirai, and MitM. They evaluated seven supervised machine learning algorithms for both binary and multiclass classification.

The ToN-IoT dataset was created in 2019 at the Australian Defence Force Academy (ADFA), University of New South Wales, Canberra, in the Cyber Range and IoT Laboratories. It comprises information gathered from interconnected devices, as well as logs from Windows and Linux systems, along with network traffic data from the system. ToN-IoT mimics complex IoT infrastructures, including virtual machines, cloud layers, edges, and physical systems. Represented in CSV format, each record in the ToN-IoT dataset is categorized as either an attack or normal behavior. Ferrag et al. [13] proposed a deep learning-based intrusion detection system using convolutional neural networks (CNNs), artificial neural networks (ANNs), and recurrent neural networks (RNNs) on the CICDDoS2019 and ToN-IoT datasets. They achieved accuracies of 95% and 99% for multiclass and binary classification on the CICDDoS2019 dataset, respectively. On the ToN-IoT dataset, they attained a multiclass classification accuracy of 98%. Kılınçer and Katar [14] conducted a multi-class classification study using the Light Gradient Boosting Machine (LGBM) classifier on the ToN_IoT dataset. The results were compared with the similar studies, and it was observed that the presented method is one of the algorithms with the highest performance for classification.

Moving on to the experimental setup, the paper details the creation of an IoT application and the execution of cyber attacks, including DDoS, HTTP Flood, SYN Flood, Port Scan, and UDP Flood, on the established IoT network. Using the necessary hardware and software, a novel IoT traffic dataset was created, namely OGU-IoT23. The dataset characteristics and the software tools used for attacks, network traffic monitoring, and machine learning algorithms are comprehensively presented.

The results section provides insights into the performance of the ML algorithms, emphasizing accuracy, F1-score, precision, recall, and test accuracy. The confusion matrix results for each algorithm, as well as the training and testing times, are thoroughly examined. Notably, Random Forest and AdaBoost algorithms exhibit superior performance across a range of metrics, making them promising candidates for IoT security applications.

In summary, this paper contributes to the understanding of the strengths and limitations of diverse ML algorithms in the context of IoT security. A new dataset, OGU-IoT23, was developed in an authentic environment, and all machine learning algorithms were equitably evaluated using this dataset. Within this research, seven distinct machine learning algorithms were assessed across five different types of attacks. The empirical evaluation, encompassing attack scenarios and diverse performance metrics, provides valuable insights for researchers and practitioners seeking effective solutions to secure IoT environments.

## Method

### Machine Learning Algorithms

In this section, a brief explanation will be given of the characteristics and functionalities of several pivotal Machine Learning (ML) algorithms employed in the field of cybersecurity, particularly within the context of an Internet of Things (IoT) application. These algorithms, namely Random Forest, AdaBoost, Decision Trees, Naive Bayes, Logistic Regression, Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN), play a crucial role in enhancing the accuracy and efficiency of classification tasks. Each algorithm will be succinctly described, outlining its unique features, strengths, and applications. The subsequent exploration of their performances, evaluated through a comprehensive experimental setup involving cyber attacks on an IoT network, aims to provide valuable insights into their effectiveness and suitability for securing IoT environments.

#### Random Forest

The Random Forest algorithm proves to be a robust and effective tool in the realm of IoT security, specifically for classification tasks [15]. Leveraging an ensemble of decision trees generated during the training phase, Random Forest excels in detecting and categorizing diverse cyber threats targeting IoT networks. Its utilization entails building numerous decision trees by employing bootstrapped samples from the training data and evaluating random subsets of features at each decision node. In IoT security scenarios, Random Forest demonstrates prowess in handling high-dimensional datasets, offering versatility in identifying various attack types, including DDoS, Port Scan, and Syn Flood attacks [16]. The algorithm's strength lies in its ability to mitigate overfitting by introducing randomness, resulting in a more resilient and accurate model. However, it is essential to acknowledge potential disadvantages, such as increased computational complexity and potential difficulties in interpreting the decision-making process due to the ensemble nature of the algorithm. Despite these considerations, the overall performance and adaptability of Random Forest make it a compelling choice for bolstering the security of IoT systems [17].

#### AdaBoost

AdaBoost, short for Adaptive Boosting, emerges as a powerful ensemble learning algorithm with significant applications in IoT security for classification tasks [18]. Its distinctive feature lies in its ability to capacity to boost performance of weak learners sequentially, assigning higher weights to misclassified instances in each iteration. In the context of IoT security, AdaBoost proves valuable for its adaptability and efficiency in recognizing and mitigating various cyber threats. By iteratively training weak learners and emphasizing misclassified instances, AdaBoost creates a strong classifier capable of accurately identifying complex attack patterns [19]. One of its primary advantages is its versatility, as it can be applied to different types of attacks within the IoT domain. However, AdaBoost's sensitivity to noisy data and outliers may pose challenges, and its performance might be affected if weak learners are too complex or if the dataset is noisy [20]. Nonetheless, AdaBoost's ability to create robust ensembles and improve overall classification accuracy makes it a compelling choice for bolstering the security of IoT systems.

**Decision Trees**

The Decision Tree algorithm stands as a versatile and interpretable tool in the landscape of IoT security, particularly for classification purposes [21]. In the realm of cybersecurity, Decision Trees recursively partition the input space based on features, creating a tree-like structure of decision nodes and leaves. In IoT security scenarios, Decision Trees excel in providing transparency into the decision-making process, aiding in the understanding of the identified attack patterns. The algorithm selects features at decision nodes to optimize classification criteria, such as Gini impurity for classification tasks [22]. While Decision Trees are intuitive and straightforward, offering a clear representation of decision logic, they may be susceptible to overfitting, especially when the trees are deep or complex. Techniques like pruning are often employed to address this concern. In the context of IoT security, Decision Trees showcase effectiveness in detecting and classifying various attacks, including DDoS, Port Scan, and SYN Flood attacks [23]. The simplicity, interpretability, and adaptability of Decision Trees render them valuable assets for fortifying the security of IoT environments.

**Naive Bayes**

In the domain of IoT security, the Naive Bayes algorithm stands out as a probabilistic classification tool widely utilized for its simplicity and efficiency. Leveraging Bayes' theorem and assuming feature independence, Naive Bayes computes the probability of a specific class given a set of features. The algorithm performs remarkably well in practice, especially in scenarios with high-dimensional datasets common in IoT applications [24]. In the context of IoT security, Naive Bayes demonstrates effectiveness in identifying and categorizing different attack types, with a notable aptitude for detecting DDoS attacks [25]. Due to its computational efficiency and straightforward implementation, Naive Bayes is well-suited for real-time applications and environments with restricted computational resources. However, the independence assumption may not always align with the intricacies of real-world datasets, posing a limitation to its application in certain contexts [26]. Despite this, Naive Bayes remains a valuable and pragmatic choice within the machine learning toolkit for IoT security applications.

**Logistic Regression**

Logistic Regression, despite its name, is a fundamental and widely applied algorithm in ML, particularly in the context of IoT security for many classification tasks [27]. It calculates the likelihood of an instance belonging to a particular class by employing the logistic function. In IoT security applications, Logistic Regression proves valuable for its simplicity, interpretability, and efficiency, particularly in cases where the relationship between features and the target variable exhibits an approximately linear nature [28]. The algorithm estimates weights for each feature by minimizing a logistic loss function, typically through iterative optimization techniques like gradient descent. The logistic function transforms the linear combination of input features and weights into a range between 0 and 1, signifying the probability of belonging to the positive class. Logistic Regression excels in scenarios where transparency into decision-making is crucial. Nonetheless, its effectiveness might be hindered when dealing with non-linear relationships between features and the target variable, potentially constraining its ability to capture intricate attack patterns. Nonetheless, Logistic Regression remains a versatile and widely-used algorithm for binary classification tasks in IoT security due to its simplicity and interpretability [29].

**Support Vector Machines**

Support Vector Machines (SVMs) emerge as potent tools in the realm of IoT security for both classification and regression tasks. SVMs demonstrate excellence in situations where the objective is to identify a hyperplane that effectively separates data points belonging to different classes in the feature space. In IoT security applications, SVMs play a pivotal role in delineating complex decision boundaries, making them effective in scenarios with intricate attack patterns [30]. The algorithm aims to optimize the margin, which is defined as the gap between the hyperplane and the nearest data points of each class, thus promoting resilient generalization to unseen data. SVMs showcase resilience against overfitting, contributing to their suitability for diverse IoT security environments [31]. However, the computational complexity of SVMs may pose challenges in resource-constrained IoT devices. Additionally, the choice of an appropriate kernel function becomes crucial, impacting the algorithm's performance. SVMs are widely adopted in IoT security for tasks like image classification, text categorization, and bioinformatics, owing to their capability to handle high-dimensional data and discern intricate attack signatures [32].

**K-Nearest Neighbors**

K-Nearest Neighbors (k-NN) stands as a straightforward yet effective algorithm in the landscape of IoT security, applicable to both classification and regression tasks [33]. Functioning on the principle of proximity, k-NN makes predictions for new instances based on the majority class or the average value of the k-nearest neighbors in the feature space. In IoT security applications, k-NN offers simplicity and ease of interpretation, making it suitable for scenarios where transparency and simplicity are valued. Nonetheless, its performance sensitivity to the chosen parameter "k" and the employed distance metric, along with the trade-off between computational cost and accuracy, should be thoughtfully taken into account. k-NN is particularly useful in IoT security applications where interpretability and simplicity are prioritized, and where a balance between computational cost and accuracy is acceptable [34].

## Performance metrics

### Confusion Matrix

The confusion matrix is a fundamental and versatile performance metric used primarily in classification tasks. It furnishes a comprehensive breakdown of the model's predictions, contrasting them with the actual class labels. (Table 1). The confusion matrix consists of four main components:

*True Positive* (TP): Corresponds to the instances accurately predicted as positive.

*True Negative* (TN): Denotes instances accurately predicted as negative.

*False Positive* (FP): Signifies instances inaccurately predicted as positive (Type I error).

*False Negative* (FN): Represents instances inaccurately predicted as negative (Type II error).

**Table 1. Confusion Matrix**

| | | **Predicted Class** | |
|---|---|---|---|
| | | **Positive** | **Negative** |
| **Actual Class** | **Positive** | TP | FN |
| | **Negative** | FP | TN |

### Accuracy

Accuracy is the ratio of correctly classified instances to the total number of instances (Eq. 1).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \qquad (1)$$

### Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It focuses on the accuracy of positive predictions (Eq. 2).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad (2)$$

### Recall (Sensitivity)

Recall, also referred to as Sensitivity, quantifies the proportion of correctly predicted positive observations relative to all observations within the actual class. It assesses the model's capacity to identify all pertinent instances (Eq. 3).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad (3)$$

### F1-Score

The F1-Score is the harmonic mean of precision and recall (Eq. 4).

$$\text{F1Skor} = 2 \text{ x } \left(\frac{\text{Precision x Recall}}{\text{Precision} + \text{Recall}}\right) \qquad (4)$$

## Experimental Setup

In this study, an IoT application has been designed using NodeMCU. To create the IoT application, an experimental environment has been established through communication between NodeMCU and the ThingSpeak cloud platform [35]. In this section, firstly, the hardware components and tools utilized in the developed IoT application are described. Subsequently, attention is given to cyber attacks conducted on the IoT application and the steps involved in creating the dataset.

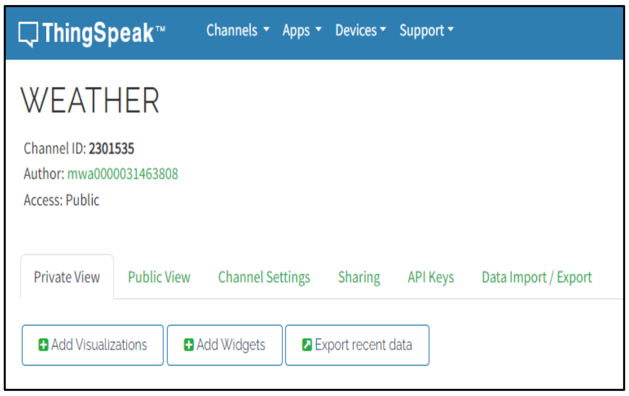
### Hardware

**NodeMCU Development Board:** NodeMCU is an open-source electronic circuit development board equipped with the ESP8266 Wi-Fi module (Figure 1). Its capability for easy internet connectivity has led to its extensive use in IoT projects requiring remote control and internet access. Its low power consumption makes it a preferred choice for IoT projects. NodeMCU can be programmed through the Arduino IDE. Power for the NodeMCU device is supplied through the built-in Micro USB connector. NodeMCU features 32 Kb RAM, 80 Kb DRAM, and 200 Kb Flash Memory (Parihar, 2019). In this study, communication with the thingspeak.com cloud platform is established using NodeMCU. NodeMCU is programmed using the Arduino IDE, and the necessary libraries for communication with the thingspeak.com cloud platform are installed and programmed.

**Figure 1.** NodeMCU with ESP8266 Wi-Fi module

**Arduino IDE:** Introduced by Arduino, Arduino IDE is software utilized for editing, compiling, and uploading code onto Arduino devices. It operates on the Arduino Java platform, making it easily accessible across different operating systems. In this study, software libraries from Arduino IDE, specifically the "ESP8266Wifi.h" and "ESP8266HTTPClient.h" libraries, have been employed.

**ThingSpeak:** ThingSpeak is an open-source and IoT-based platform that communicates over the internet using MQTT and HTTP protocols, storing data collected from objects and presenting it to users [35]. Through this platform, real-time tracking can be performed, allowing for the addition and removal of data, as well as analysis and control processes [36]. It is available for free use for one year and enables approximately 3 million messages to be sent. Data can be sent, recorded, and visualized every 15 seconds. In this study, ThingSpeak is utilized via NodeMCU to record and visualize data retrieved over the internet. To send data, it is necessary to create a channel after registering and logging into this web application (Figure 2).



WEATHER

Channel ID: 2301535
Author: mwa0000031463808
Access: Public

**Figure 2.** Creating a channel in ThingSpeak

**IoT Experimental Environment**

In this section, details regarding the communication among the devices used in the setup of the IoT experimental environment and the configurations are shared. As depicted in Figure 3, a experimental environment has been established in this study.

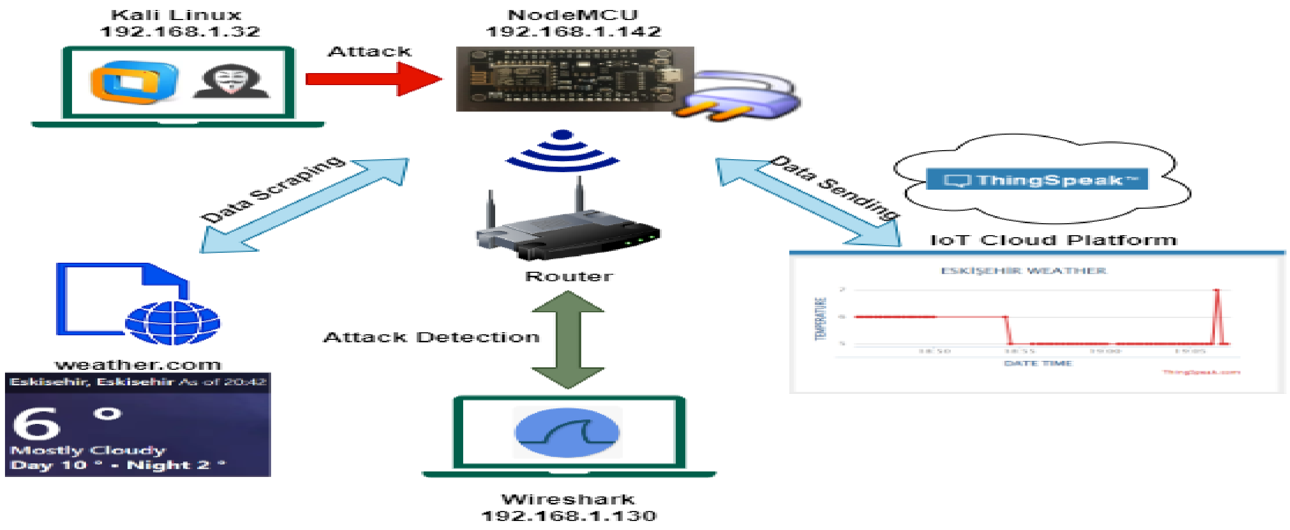


**Figure 3**. IoT experimental environment

In this established system, the IoT device with the IP address 192.168.1.xyz communicates wirelessly with the router in our network. The IoT device initially connects to the weather.com webpage and extracts real-time weather information for the city of Eskişehir through web scraping. Subsequently, it promptly transmits this acquired data to the Thingspeak cloud platform. The sent data can be viewed by users possessing the necessary API credentials. While the system continues its normal operation, diverse attacks are executed from the Kali Linux machine with the IP address 192.168.1.pq towards the IoT device. Throughout the ongoing attacks, network traffic is recorded by the machine with the IP address 192.168.1.abc, which is equipped with Wireshark. The process of recording under various attacks is carried out sequentially. The network traffic in the established platform is saved in CSV format. The steps of detecting and classifying the attacks are illustrated in Figure 4.

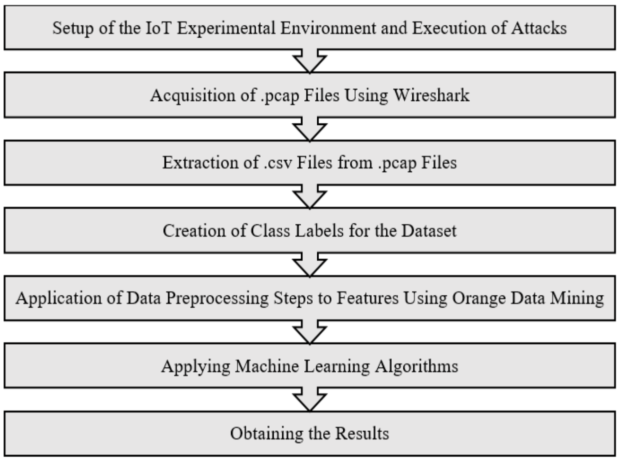

Setup of the IoT Experimental Environment and Execution of Attacks
Acquisition of .pcap Files Using Wireshark
Extraction of .csv Files from .pcap Files
Creation of Class Labels for the Dataset
Application of Data Preprocessing Steps to Features Using Orange Data Mining
Applying Machine Learning Algorithms
Obtaining the Results

**Figure 4.** Flowchart of detection and classification of attacks.

**Software**

In this study, various attacks were conducted on the designed IoT network, and these attacks were monitored to record network data. Table 2 provides information on the attack tools, network traffic monitoring tools, cloud platform interface, and the software where machine learning algorithms were executed in this study.

**Table 2.** Utilized Software and Their Purposes

| Software | Purpose |
|---|---|
| **Wireshark** | Network Traffic Monitoring |
| **LOIC** | Attack Tool |
| **Hping3** | Attack Tool |
| **Orange Data Mining** | Machine Learning Algorithms |
| **Thingspeak** | IoT Cloud Platform |

## Dataset

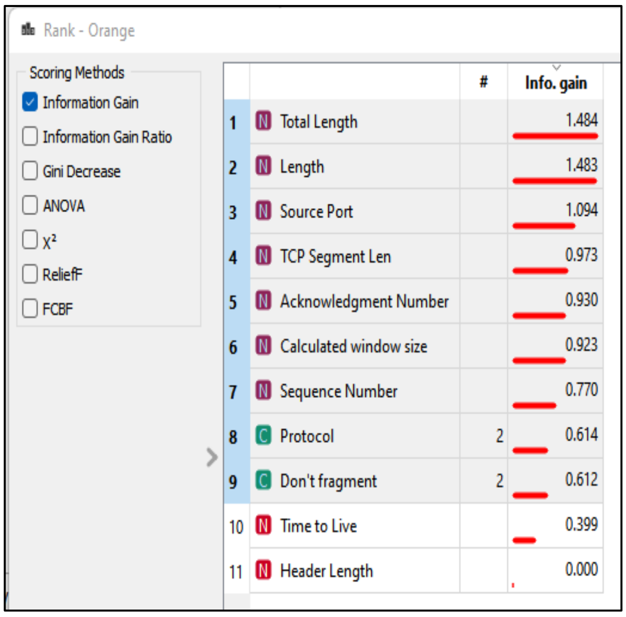
In this study, the aim is to create a balanced dataset encompassing attacks on IoT devices. For the desired dataset, various attacks, including DDoS, HTTP Flood, Port Scan, SYN Flood, and UDP Flood, were performed on an IoT device in a laboratory environment. Kali Linux was used to carry out the attacks and to detect them afterward using the Pcap files captured by Wireshark.

**Table 3.** Dataset Characteristics

| No | Features | Area | Explanation |
|---|---|---|---|
| **1** | Total Length | ip.len | Total Length |
| **2** | Don't Fragment | ip.flags.df | Fragmentation Value |
| **3** | Time To Live | ip.ttl | Packet Time-to-Live |
| **4** | Protocol | ip.proto | IP Protocol Type |
| **5** | Calculated Window Size | tcp.window_size | TCP Packet Window Size |
| **6** | Acknowledgment Number | tcp.ack | TCP Packet Acknowledgment Number |
| **7** | Sequence Number | tcp.seq | TCP Packet Sequence Number |
| **8** | Length | bytes | Packet Length |
| **9** | TCP Segment Len | tcp.len | TCP Packet Segment Length |
| **10** | Source Port | tcp.srcport | TCP Packet Source Port |
| **11** | Header Length | ip.hdr_len | IP Packet Header Length |
| **12** | Class | Class | Class Label |

The dataset, OGU-IoT23 comprises a total of 116,819 instances, featuring six different classes, including normal packets. Since the dataset is sufficiently large and diverse, a fixed proportion data sampling was used, allocating 80% for training and 20% for testing. This method leveraged the dataset's richness for robust model evaluation. Moreover, to address potential overfitting, we employed regularization techniques and validation procedures during training. For the calculation of performance metrics, we utilized 5-fold Cross-Validation, ensuring a comprehensive assessment of the model's performance across different subsets of the data.

As seen in Table 3, our dataset is composed of 11 features and consists of six classes: DDoS, HTTP Flood, Normal, Port Scan, SYN Flood, and UDP Flood. The dataset includes 20,001 examples for the DDoS class, 19,970 examples for the HTTP Flood class, 18,886 examples for the Normal class, 18,854 examples for the Port Scan class, 20,707 examples for the SYN Flood class, and 18,401 examples for the UDP Flood class. In order to expedite the classification process, 9 out of the 11 features were selected using the Information Gain algorithm. The weight rankings of the Information Gain algorithm are illustrated in Figure 5.



Rank - Orange

Scoring Methods
- [✓] Information Gain
- [ ] Information Gain Ratio
- [ ] Gini Decrease
- [ ] ANOVA
- [ ] $\chi^2$
- [ ] ReliefF
- [ ] FCBF

| | | # | Info. gain |
|---|---|---|---|
| 1 | N Total Length | | 1.484 |
| 2 | N Length | | 1.483 |
| 3 | N Source Port | | 1.094 |
| 4 | N TCP Segment Len | | 0.973 |
| 5 | N Acknowledgment Number | | 0.930 |
| 6 | N Calculated window size | | 0.923 |
| 7 | N Sequence Number | | 0.770 |
| 8 | C Protocol | 2 | 0.614 |
| 9 | C Don't fragment | 2 | 0.612 |
| 10 | N Time to Live | | 0.399 |
| 11 | N Header Length | | 0.000 |

**Figure 5.** Features Selected with Information Gain

## Attacks

In this study, DDoS, HTTP Flood, SYN Flood, Port Scan, and UDP Flood attacks were conducted, and attempts were made to detect them using machine learning algorithms determined by monitoring network traffic.

**DDoS Attack**

A DDoS attack involves a malevolent effort to disrupt a targeted device by overwhelming it with a load far beyond normal limits [37]. DDoS attacks are executed by utilizing multiple computer systems as sources of attack traffic. Such attacks prevent regular traffic from reaching its intended destination [38]. In this study, the Hping3 tool was employed to conduct a DDoS attack on the IoT device. As depicted in Figure 6, the data loss during attack periods, as observed from the ThingSpeak platform, aligns temporally with the Wireshark I/O Graphs.

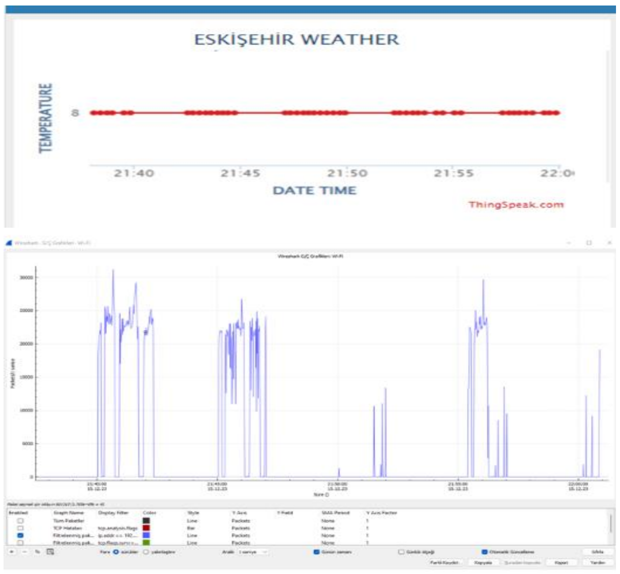


**Figure 6.** ThingSpeak and Wireshark Graphs During DDoS Attack

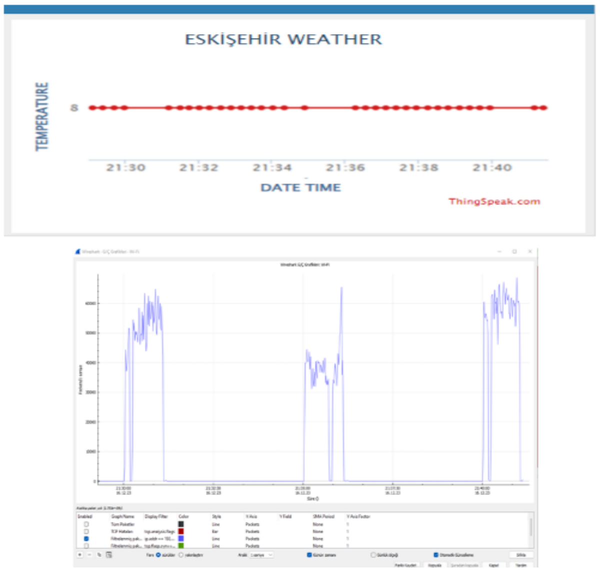


**Figure 7.** ThingSpeak and Wireshark Graphs During SYN Flood Attack

## HTTP Flood Attack

HTTP flood attack involves overwhelming a device by inundating it with a substantial volume of HTTP requests, depleting the device's resources and rendering it non-operational [39]. These requests are often fake or deceptive and are made to exhaust the device's resources. In this study, the IoT device sending data to the ThingSpeak platform was subjected to HTTP Flood attacks at intervals using the LOIC tool.

## SYN Flood Attack

A SYN Flood attack is executed to deplete the resources of the target device and render it inoperable [40]. During a SYN Flood attack, the assailant sends numerous SYN messages to the target device. For each SYN received, the device creates an entry in its connection table and responds with a SYN-ACK message. The attacker either does not send an ACK message or frequently provides an incorrect IP address in SYN packets, causing the target device not to receive SYN-ACK responses. As the attacker persists in sending SYN messages, the connection table of the target device becomes full, and the device cannot respond to any connection requests. Exhausting all resources, the target device generates a denial-of-service and cannot establish connections with clients [41]. In this study, the IoT device sending data to the ThingSpeak platform was subjected to SYN Flood attacks at intervals. The impact of the attack can be traced in the ThingSpeak platform's provided widgets and Wireshark I/O graphs. As observed in Figure 7, data transmission from the IoT device to the ThingSpeak platform is disrupted during three different time intervals when the attack is executed.

## Port Scan Attack

A Port Scan attack aims to discover open ports and exploit a known security vulnerability by sending requests to device port addresses [42]. In this study, the IoT device sending weather information to the ThingSpeak platform was subjected to a port scan attack using the Hping3 tool. The adverse effects of the Port Scan attack can be observed in the widgets provided by the ThingSpeak platform and in the Wireshark I/O graphs. As illustrated in Figure 8, the transmission of data from the IoT device to the ThingSpeak platform experiences disruptions during three distinct time intervals when the attack is carried out.

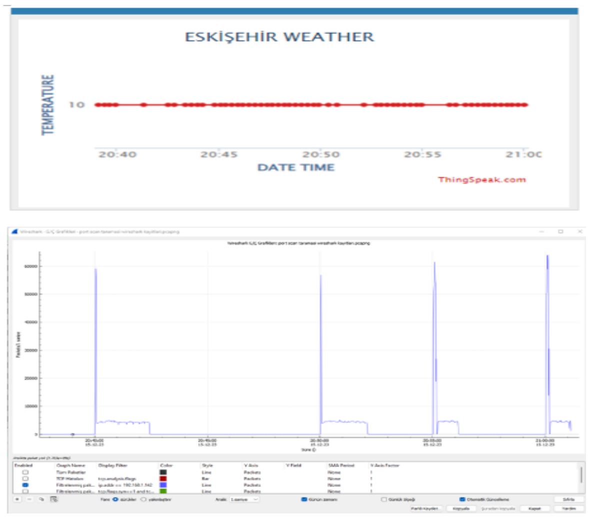


**Figure 8.** ThingSpeak and Wireshark Graphs During Port Scan Attack

## UDP Flood Attack

A UDP Flood attack is a kind of attack that aims to render the target device inoperable by consuming its network bandwidth with UDP packets. UDP Flood attacks can target the port or IP addresses of a device within a network, either randomly or specifically [43]. In this study, the IoT device sending data to the ThingSpeak platform was subjected to a UDP Flood attack using the LOIC tool.

## Results

In this study, the dataset collected using the Orange Data Mining application was trained. For the training set, 80% of the data was utilized, while 20% was reserved for the test set. Subsequently, feature selection was performed using information gain from the dataset. A model was then created using seven different classification algorithms: AdaBoost, k-Nearest Neighbors (k-NN), Logistic Regression, Naive Bayes, Random Forest, Support Vector Machine (SVM), and Decision Tree. The structure of the model is illustrated in Figure 9.

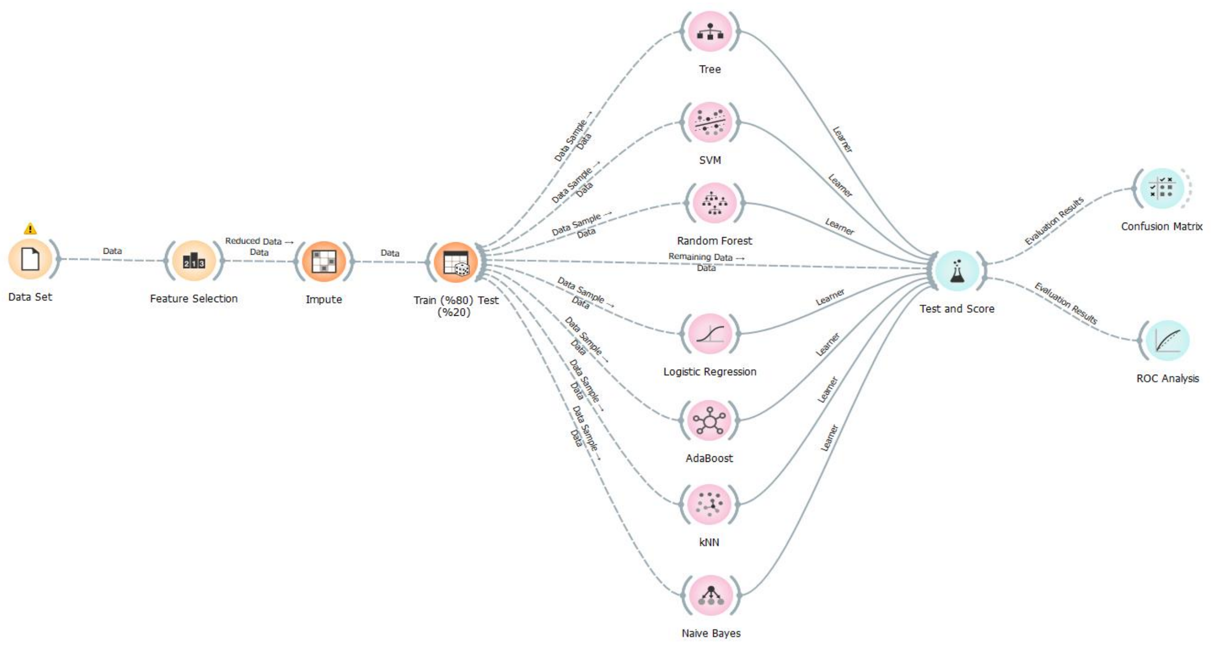


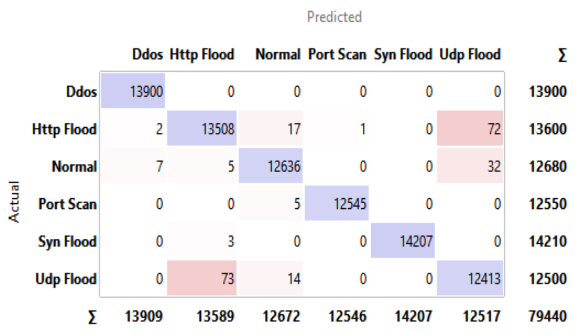
**Figure 9.** Machine Learning Model Workflow Diagram

The performance metrics of the created model have been calculated. Among the algorithms, the Random Forest algorithm yields the highest accuracy score compared to the other six algorithms, with a score of 99%. The lowest accuracy score is obtained with the Logistic Regression classification algorithm, achieving a percentage accuracy of 51%. The results are presented in Table 4.

**Table 4.** Comparison of ML Algorithms

| Model | Accuracy | F1 Score | Precision | Recall |
|---|---|---|---|---|
| **AdaBoost** | 0.997 | 0.997 | 0.997 | 0.997 |
| **k-NN** | 0.576 | 0.575 | 0.591 | 0.576 |
| **Logistic Regression** | 0.515 | 0.458 | 0.468 | 0.515 |
| **Naive Bayes** | 0.889 | 0.885 | 0.920 | 0.889 |
| **Random Forest** | 0.997 | 0.997 | 0.997 | 0.997 |
| **SVM** | 0.560 | 0.567 | 0.613 | 0.560 |
| **Decision Tree** | 0.990 | 0.99 | 0.990 | 0.990 |

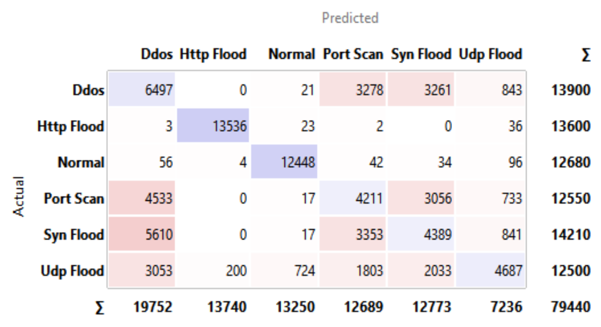
The confusion matrix results of the models were obtained separately for each model. In Figure 10, the confusion matrix results of the AdaBoost algorithm are presented. It is observed that the AdaBoost algorithm successfully detects Port Scan and SYN Flood attacks more effectively compared to other attacks. According to the confusion matrix results, the AdaBoost algorithm is identified as one of the successful detection algorithms. In this experiment, number of estimators was set to 50 and linear regression loss function was used.


Predicted

| Actual | Ddos | Http Flood | Normal | Port Scan | Syn Flood | Udp Flood | Σ |
|---|---|---|---|---|---|---|---|
| Ddos | 13900 | 0 | 0 | 0 | 0 | 0 | 13900 |
| Http Flood | 2 | 13508 | 17 | 1 | 0 | 72 | 13600 |
| Normal | 7 | 5 | 12636 | 0 | 0 | 32 | 12680 |
| Port Scan | 0 | 0 | 5 | 12545 | 0 | 0 | 12550 |
| Syn Flood | 0 | 3 | 0 | 0 | 14207 | 0 | 14210 |
| Udp Flood | 0 | 73 | 14 | 0 | 0 | 12413 | 12500 |
| Σ | 13909 | 13589 | 12672 | 12546 | 14207 | 12517 | 79440 |

**Figure 10.** The confusion matrix of the AdaBoost algorithm.

In Figure 11, the confusion matrix results of the k-NN algorithm are depicted. It is observed that the k-NN algorithm successfully detects HTTP Flood attacks more effectively compared to other attacks. For this test, k was taken as 5, Euclidean used as metric type and Uniform used as weight.

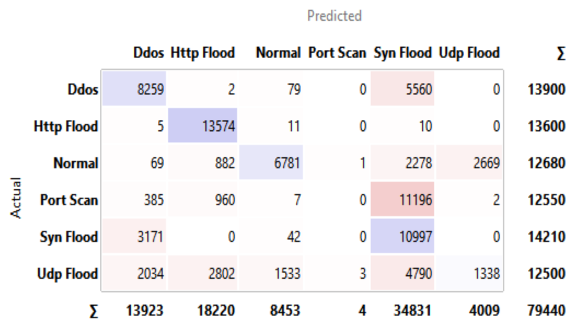


Predicted

| Actual | | Ddos | Http Flood | Normal | Port Scan | Syn Flood | Udp Flood | Σ |
|---|---|---|---|---|---|---|---|---|
| | Ddos | 6497 | 0 | 21 | 3278 | 3261 | 843 | 13900 |
| | Http Flood | 3 | 13536 | 23 | 2 | 0 | 36 | 13600 |
| | Normal | 56 | 4 | 12448 | 42 | 34 | 96 | 12680 |
| | Port Scan | 4533 | 0 | 17 | 4211 | 3056 | 733 | 12550 |
| | Syn Flood | 5610 | 0 | 17 | 3353 | 4389 | 841 | 14210 |
| | Udp Flood | 3053 | 200 | 724 | 1803 | 2033 | 4687 | 12500 |
| | Σ | 19752 | 13740 | 13250 | 12689 | 12773 | 7236 | 79440 |

**Figure 11.** The confusion matrix of the k-NN algorithm.

In Figure 12, the confusion matrix results of the Logistic Regression algorithm are presented. Similar to the k-NN algorithm, it is noted that the Logistic Regression algorithm successfully detects HTTP Flood attacks more effectively compared to other attacks. In this experiment, Ridge (L2) was applied as regularization and C (strength) was set to 1.

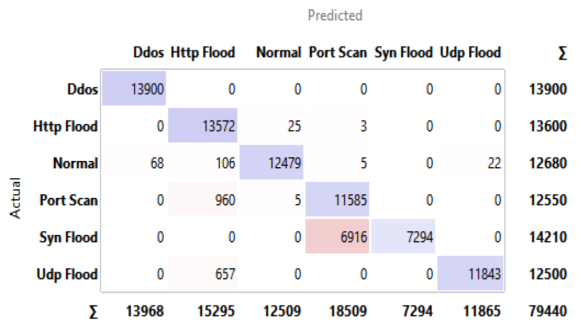


Predicted

| Actual | | Ddos | Http Flood | Normal | Port Scan | Syn Flood | Udp Flood | Σ |
|---|---|---|---|---|---|---|---|---|
| | Ddos | 8259 | 2 | 79 | 0 | 5560 | 0 | 13900 |
| | Http Flood | 5 | 13574 | 11 | 0 | 10 | 0 | 13600 |
| | Normal | 69 | 882 | 6781 | 1 | 2278 | 2669 | 12680 |
| | Port Scan | 385 | 960 | 7 | 0 | 11196 | 2 | 12550 |
| | Syn Flood | 3171 | 0 | 42 | 0 | 10997 | 0 | 14210 |
| | Udp Flood | 2034 | 2802 | 1533 | 3 | 4790 | 1338 | 12500 |
| | Σ | 13923 | 18220 | 8453 | 4 | 34831 | 4009 | 79440 |

**Figure 12.** The confusion matrix of the Logistic Regression algorithm.

In Figure 13, the confusion matrix results of the Naive Bayes algorithm are displayed. It is observed that the Naive Bayes algorithm successfully detects DDoS attacks more effectively compared to other attacks. Here, Laplace method was selected as smoothing parameter and Information Gain was used for Feature Selection.

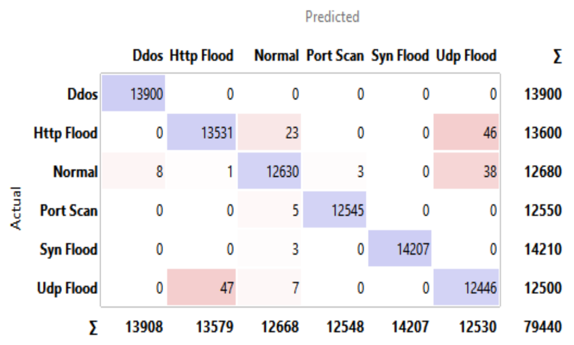


Predicted

| Actual | | Ddos | Http Flood | Normal | Port Scan | Syn Flood | Udp Flood | Σ |
|---|---|---|---|---|---|---|---|---|
| | Ddos | 13900 | 0 | 0 | 0 | 0 | 0 | 13900 |
| | Http Flood | 0 | 13572 | 25 | 3 | 0 | 0 | 13600 |
| | Normal | 68 | 106 | 12479 | 5 | 0 | 22 | 12680 |
| | Port Scan | 0 | 960 | 5 | 11585 | 0 | 0 | 12550 |
| | Syn Flood | 0 | 0 | 0 | 6916 | 7294 | 0 | 14210 |
| | Udp Flood | 0 | 657 | 0 | 0 | 0 | 11843 | 12500 |
| | Σ | 13968 | 15295 | 12509 | 18509 | 7294 | 11865 | 79440 |

**Figure 13.** The confusion matrix of the Naive Bayes algorithm.

In Figure 14, the confusion matrix results of the Random Forest algorithm are presented. The Random Forest algorithm is observed to successfully detect DDoS, Port Scan, and Syn Flood attacks more effectively compared to other attacks. According to the confusion matrix results, the Random Forest is identified as one of the successful detection algorithms. Here, number of trees was set to 7 and subsets were not split if less than 5.

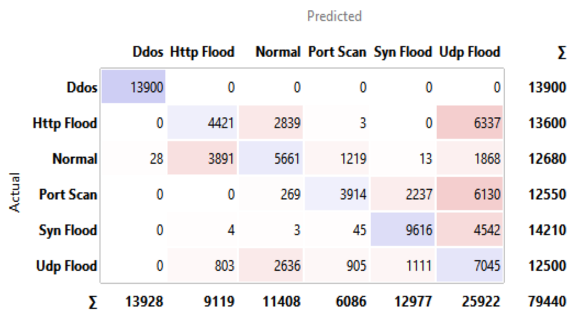


Predicted

| Actual | | Ddos | Http Flood | Normal | Port Scan | Syn Flood | Udp Flood | Σ |
|---|---|---|---|---|---|---|---|---|
| | Ddos | 13900 | 0 | 0 | 0 | 0 | 0 | 13900 |
| | Http Flood | 0 | 13531 | 23 | 0 | 0 | 46 | 13600 |
| | Normal | 8 | 1 | 12630 | 3 | 0 | 38 | 12680 |
| | Port Scan | 0 | 0 | 5 | 12545 | 0 | 0 | 12550 |
| | Syn Flood | 0 | 0 | 3 | 0 | 14207 | 0 | 14210 |
| | Udp Flood | 0 | 47 | 7 | 0 | 0 | 12446 | 12500 |
| | Σ | 13908 | 13579 | 12668 | 12548 | 14207 | 12530 | 79440 |

**Figure 14.** The confusion matrix of the Random Forest algorithm.

In Figure 15, the confusion matrix results of the SVM algorithm are depicted. Similar to the Naive Bayes algorithm, it is observed that the SVM algorithm successfully detects DDoS attacks more effectively compared to other attacks. In this experiment, RBF was used as Kernel and iteration limit was set to 100.

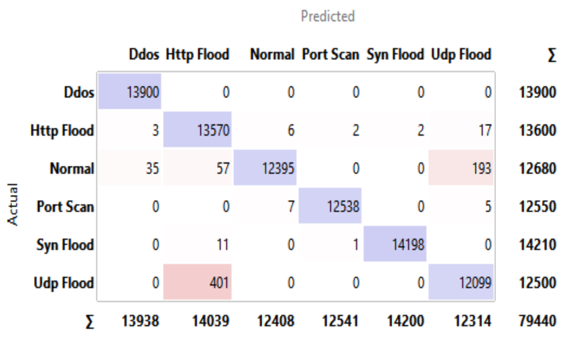


Predicted

| Actual | | Ddos | Http Flood | Normal | Port Scan | Syn Flood | Udp Flood | Σ |
|---|---|---|---|---|---|---|---|---|
| | Ddos | 13900 | 0 | 0 | 0 | 0 | 0 | 13900 |
| | Http Flood | 0 | 4421 | 2839 | 3 | 0 | 6337 | 13600 |
| | Normal | 28 | 3891 | 5661 | 1219 | 13 | 1868 | 12680 |
| | Port Scan | 0 | 0 | 269 | 3914 | 2237 | 6130 | 12550 |
| | Syn Flood | 0 | 4 | 3 | 45 | 9616 | 4542 | 14210 |
| | Udp Flood | 0 | 803 | 2636 | 905 | 1111 | 7045 | 12500 |
| | Σ | 13928 | 9119 | 11408 | 6086 | 12977 | 25922 | 79440 |

**Figure 15.** The confusion matrix of the SVM algorithm.

In Figure 16, the confusion matrix results of the Decision Tree algorithm are presented. The Decision Tree algorithm is observed to successfully detect DDoS, Port Scan, and SYN Flood attacks more effectively compared to other attacks. According to the confusion matrix results, the Decision Tree algorithm is identified as one of the successful detection algorithms. For this test, minimum number of instances in leaves was set to 5 and the limit of the maximal tree dept was chosen as 100.



Predicted

| Actual | | Ddos | Http Flood | Normal | Port Scan | Syn Flood | Udp Flood | Σ |
|---|---|---|---|---|---|---|---|---|
| | Ddos | 13900 | 0 | 0 | 0 | 0 | 0 | 13900 |
| | Http Flood | 3 | 13570 | 6 | 2 | 2 | 17 | 13600 |
| | Normal | 35 | 57 | 12395 | 0 | 0 | 193 | 12680 |
| | Port Scan | 0 | 0 | 7 | 12538 | 0 | 5 | 12550 |
| | Syn Flood | 0 | 11 | 0 | 1 | 14198 | 0 | 14210 |
| | Udp Flood | 0 | 401 | 0 | 0 | 0 | 12099 | 12500 |
| | Σ | 13938 | 14039 | 12408 | 12541 | 14200 | 12314 | 79440 |

**Figure 17.** The confusion matrix of the Decision Tree algorithm.

In this study, the calculated training and testing times during the classification process are presented in Table 5. When evaluated alongside other performance metrics, particularly the Naïve Bayes and Decision Tree algorithms exhibit notably faster training and testing times compared to other algorithms.

As a summary of the some studies along with their datasets and this study is briefly presented in Table 6. By examining these studies collectively, readers gain insights into the performance and applicability of intrusion detection systems across different dataset environments and classification types, thus aiding in the selection and evaluation of suitable methodologies for securing networked systems against cyber threats.

**Table 5.** The training and testing times of the algorithms.

| Model | Training (sec) | Test (sec) |
|---|---|---|
| **AdaBoost** | 22.420 | 2.164 |
| **k-NN** | 0.373 | 2.287 |
| **Logistic Regression** | 2.431 | 0.097 |
| **Naive Bayes** | 0.222 | 0.058 |
| **Random Forest** | 0.652 | 0.163 |
| **SVM** | 20.315 | 19.285 |
| **Decision Tree** | 0.307 | 0.017 |

**Table 6.** Comparison of Intrusion Detection Studies: Datasets, Classifications, Dataset Types, and Accuracies

| Research | Dataset | Classification | Dataset Type | Accuracy |
|---|---|---|---|---|
| (Farahnakian & Heikkonen, 2018) | KDD CUP 99 | Binary | Simulation | 96% |
| (Khalvati, Keshtgary, & Rikhtegar, 2018) | KDD CUP 99 | Multi-Class | Simulation | 91% |
| (Ullah & Mahmoud, 2020) | IoTID20 | Multi-Class<br>Binary | Real Environment | 100%<br>100% |
| (Ferrag, Shu , Djallel , & Choo, 2021) | CICDDoS2019 | Multi-Class<br>Binary | Simulation | 95%<br>99% |
| (Kılınçer, İ. F., & Katar, O. 2023) | ToN-IoT | Multi-Class | Simulation | 99% |
| Our Work | OGU-IoT23 (our novel dataset) | Multi-Class | Real Environment | 99% |

## Discussion

This study extensively explores the performance of prominent machine learning (ML) algorithms in securing Internet of Things (IoT) applications, focusing on Random Forest, AdaBoost, Decision Trees, Naive Bayes, Logistic Regression, Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN). Through a comprehensive experimental setup involving cyber attacks on an IoT network, the algorithms are assessed for their strengths and weaknesses.

The experimental setup involves the creation of an IoT application using NodeMCU and the execution of various cyber attacks, including DDoS, HTTP Flood, SYN Flood, Port Scan, and UDP Flood. A novel dataset named OGU-IoT23 was generated within a real-world environment, and all machine learning algorithms were fairly assessed using this dataset. The dataset characteristics, hardware components, and tools used for attacks, network traffic monitoring, and ML algorithms are comprehensively presented.

Notably, the performance discrepancies observed among the ML algorithms underscore the necessity of selecting suitable algorithms tailored to the specific requirements of IoT security. While Random Forest and AdaBoost exhibit remarkable accuracy across diverse attack scenarios, algorithms like Logistic Regression and k-Nearest Neighbors (k-NN) demonstrate comparatively lower performance metrics, reflecting the varied strengths and weaknesses inherent in each algorithm. The ensemble nature of Random Forest and AdaBoost enhances their robustness against a wide range of attack vectors, while simpler algorithms like Logistic Regression and Naive Bayes may struggle to capture the complexities of IoT network traffic.

It's imperative to assess the robustness and generalization capabilities of ML algorithms beyond the experimental setup, necessitating further validation across diverse datasets and real-world IoT deployment scenarios. Additionally, the choice of dataset significantly influences algorithm performance and generalizability. Although the OGU-IoT23 dataset used in this study reflects realistic IoT network traffic, future research could explore the transferability of ML models trained on diverse datasets to enhance their applicability across different IoT deployment environments and attack landscapes. Integrating ML-based intrusion detection systems with broader cybersecurity frameworks is essential for effective IoT security, ensuring a comprehensive defense strategy against evolving cyber threats.

## Conclusion

This study presents a comprehensive exploration of the performance of prominent machine learning (ML) algorithms in securing Internet of Things (IoT) applications. Through rigorous experimentation involving cyber attacks on an IoT network, the study assesses the strengths and weaknesses of various ML algorithms. Each algorithm is introduced with its unique characteristics and applications, highlighting their suitability for different IoT security contexts. Results demonstrate Random Forest and AdaBoost as top performers, showcasing superior accuracy and effectiveness across multiple attack scenarios. The study underscores the importance of selecting appropriate algorithms based on specific IoT application requirements and attack scenarios. It provides valuable insights for researchers and practitioners seeking effective solutions to secure IoT environments, considering both algorithmic performance and computational efficiency. By understanding the strengths and limitations of different ML algorithms, developers can make informed decisions when designing and deploying intrusion detection systems for IoT security, ultimately contributing to the advancement of IoT security practices.

## Ethics committee approval

"There is no need to obtain permission from the ethics committee for the article prepared"

## Conflict of Interest

"There is no conflict of interest with any person / institution in the article prepared"

## Authors' Contributions

First Author: Acquisition of data, Analysis and interpretation of data

Second Author: Study conception and design, Drafting of manuscript, Critical revision

## References

[1] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Comput. Sci.*, vol. 2, no. 3, p. 160, May 2021, doi: 10.1007/s42979-021-00592-x.

[2] B. Mahesh, "Machine Learning Algorithms - A Review," *Int. J. Sci. Res.*, vol. 9, no. 1, pp. 381–386, 2020, doi: 10.21275/ART20203995.

[3] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[4] Y. CAO, Q.-G. MIAO, J.-C. LIU, and L. GAO, "Advance and Prospects of AdaBoost Algorithm," *Acta Autom. Sin.*, vol. 39, no. 6, pp. 745–758, Jun. 2013, doi: 10.1016/S1874-1029(13)60052-X.

[5] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," *J. Appl. Sci. Technol. Trends*, vol. 2, no. 01, pp. 20–28, Mar. 2021, doi: 10.38094/jastt20165.

[6] A. Yasar and M. M. Saritas, "Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification," *Int. J. Intell. Syst. Appl. Eng.*, vol. 7, no. 2, pp. 88–91, 2019, doi: 10.18201/ijisae.2019252786.

[7] R. D. Joshi and C. K. Dhakal, "Predicting Type 2 Diabetes Using Logistic Regression and Machine Learning Approaches," *Int. J. Environ. Res. Public Health*, vol. 18, no. 14, p. 7346, Jul. 2021, doi: 10.3390/ijerph18147346.

[8] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, Sep. 2020, doi: 10.1016/j.neucom.2019.10.118.

[9] P. Cunningham and S. J. Delany, "k-Nearest Neighbour Classifiers - A Tutorial," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–25, Jul. 2022, doi: 10.1145/3459665.

[10] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2018, pp. 178–183, doi: 10.23919/ICACT.2018.8323688.

[11] L. Khalvati, M. Keshtgary, and N. Rikhtegar, "Intrusion Detection based on a Novel Hybrid Learning Approach," *J. AI Data Min.*, vol. 6, no. 1, pp. 157–162, 2018, doi: 10.22044/jadm.2017.979.

[12] I. Ullah and Q. H. Mahmoud, "A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks," in *Advances in Artificial Intelligence*, 2020, pp. 508–520.

[13] M. A. Ferrag, L. Shu, H. Djallel, and K.-K. R. Choo, "Deep Learning-Based Intrusion Detection for Distributed Denial of Service Attack in Agriculture 4.0," *Electronics*, vol. 10, no. 11, p. 1257, May 2021, doi: 10.3390/electronics10111257.

[14] İ. F. KILINÇER and O. KATAR, "A new Intrusion Detection System for Secured IoT/IIoT Networks based on LGBM," *Gazi Üniversitesi Fen Bilim. Derg. Part C Tasarım ve Teknol.*, vol. 11, no. 2, pp. 321–328, Jun. 2023, doi: 10.29109/gujsc.1173286.

[15] S. K. Lakshmanaprabu, K. Shankar, M. Ilayaraja, A. W. Nasir, V. Vijayakumar, and N. Chilamkurti, "Random forest for big data classification in the internet of things using optimal features," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 10, pp. 2609–2618, Oct. 2019, doi: 10.1007/s13042-018-00916-z.

[16] P. Kumar, H. Bagga, B. S. Netam, and V. Uduthalapally, "SAD-IoT: Security Analysis of DDoS Attacks in IoT Networks," *Wirel. Pers. Commun.*, vol. 122, no. 1, pp. 87–108, Jan. 2022, doi: 10.1007/s11277-021-08890-6.

[17] W. Feng, C. Ma, G. Zhao, and R. Zhang, "FSRF:An Improved Random Forest for Classification," in *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications( AEECA)*, Aug. 2020, pp. 173–178, doi: 10.1109/AEECA49918.2020.9213456.

[18] F. Wang, D. Jiang, H. Wen, and H. Song, "Adaboost-based security level classification of mobile intelligent terminals," *J. Supercomput.*, vol. 75, no. 11, pp. 7460–7478, Nov. 2019, doi: 10.1007/s11227-019-02954-y.

[19] S. Rachmadi, S. Mandala, and D. Oktaria, "Detection of DoS Attack using AdaBoost Algorithm on IoT System," in *2021 International Conference on Data Science and Its Applications (ICoDSA)*, Oct. 2021, pp. 28–33, doi: 10.1109/ICoDSA53588.2021.9617545.

[20] E. Nazarenko, V. Varkentin, and T. Polyakova, "Features of Application of Machine Learning Methods for Classification of Network Traffic (Features, Advantages, Disadvantages)," in *2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, Oct. 2019, pp. 1–5, doi: 10.1109/FarEastCon.2019.8934236.

[21] M. Douiba, S. Benkirane, A. Guezzaz, and M. Azrour, "An improved anomaly detection model for IoT security using decision tree and gradient boosting," *J. Supercomput.*, vol. 79, no. 3, pp. 3392–3411, Feb. 2023, doi: 10.1007/s11227-022-04783-y.

[22] M. A. Bouke, A. Abdullah, S. H. ALshatebi, M. T. Abdullah, and H. El Atigh, "An intelligent DDoS attack detection tree-based model using Gini index feature selection method," *Microprocess. Microsyst.*, vol. 98, p. 104823, Apr. 2023, doi: 10.1016/j.micpro.2023.104823.

[23] M. Aamir, S. S. H. Rizvi, M. A. Hashmani, M. Zubair, and J. A. . Usman, "Machine Learning Classification of Port Scanning and DDoS Attacks: A Comparative Analysis," *Mehran Univ. Res. J. Eng. Technol.*, vol. 40, no. 1, pp. 215–229, Jan. 2021, doi: 10.22581/muet1982.2101.19.

[24] L. Chen and S. Wang, "Automated feature weighting in naive bayes for high-dimensional data classification," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, Oct. 2012, pp. 1243–1252, doi: 10.1145/2396761.2398426.

[25] A. Mehmood, M. Mukherjee, S. H. Ahmed, H. Song, and K. M. Malik, "NBC-MAIDS: Naïve Bayesian classification technique in multi-agent system-enriched IDS for securing IoT against DDoS attacks," *J. Supercomput.*, vol. 74, no. 10, pp. 5156–5170, Oct. 2018, doi: 10.1007/s11227-018-2413-7.

[26] J. Ren, S. D. Lee, X. Chen, B. Kao, R. Cheng, and D. Cheung, "Naive Bayes Classification of Uncertain Data," in *2009 Ninth IEEE International Conference on Data Mining*, Dec. 2009, pp. 944–949, doi: 10.1109/ICDM.2009.90.

[27] K. Prathapchandran and T. Janani, "A Trust-Based Security Model to Detect Misbehaving Nodes in Internet of Things (IoT) Environment using Logistic Regression," *J. Phys. Conf. Ser.*, vol. 1850, no. 1, p. 012031, May 2021, doi: 10.1088/1742-6596/1850/1/012031.

[28] N. K. Sahu and I. Mukherjee, "Machine Learning based anomaly detection for IoT Network: (Anomaly detection in IoT Network)," in *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)*, Jun. 2020, pp. 787–794, doi: 10.1109/ICOEI48184.2020.9142921.

[29] F. Abbasi, M. Naderan, and S. E. Alavi, "Anomaly detection in Internet of Things using feature selection and classification based on Logistic Regression and Artificial Neural Network on N-BaIoT dataset," in *2021 5th International Conference on Internet of Things and Applications (IoT)*, May 2021, pp. 1–7, doi: 10.1109/IoT52625.2021.9469605.

[30] C. Ioannou and V. Vassiliou, "Network Attack Classification in IoT Using Support Vector Machines," *J. Sens. Actuator Networks*, vol. 10, no. 3, p. 58, Aug. 2021, doi: 10.3390/jsan10030058.

[31] A. Mubarakali, K. Srinivasan, R. Mukhalid, S. C. B. Jaganathan, and N. Marina, "Security challenges in internet of things: Distributed denial of service attack detection using support vector machine-based expert systems," *Comput. Intell.*, vol. 36, no. 4, pp. 1580–1592, Nov. 2020, doi: 10.1111/coin.12293.

[32] M. Al-Akhras, M. Alawairdhi, A. Alkoudari, and S. Atawneh, "Using Machine Learning to Build a Classification Model for IoT Networks to Detect Attack Signatures," *Int. J. Comput. Networks Commun.*, vol. 12, no. 6, pp. 99–116, Nov. 2020, doi: 10.5121/ijcnc.2020.12607.

[33] M. Mohy-eddine, A. Guezzaz, S. Benkirane, and M. Azrour, "An efficient network intrusion detection model for IoT security using K-NN classifier and feature selection," *Multimed. Tools Appl.*, vol. 82, no. 15, pp. 23615–23633, Jun. 2023, doi: 10.1007/s11042-023-14795-2.

[34] Y. Liao and V. R. Vemuri, "Use of K-Nearest Neighbor classifier for intrusion detection," *Comput. Secur.*, vol. 21, no. 5, pp. 439–448, Oct. 2002, doi: 10.1016/S0167-4048(02)00514-X.

[35] M. A. Gómez Maureira, D. Oldenhof, and L.

Teernstra, “ThingSpeak – an API and Web Service for the Internet of Things,” *World Wide Web*. 2014, [Online]. Available: https://staas.home.xs4all.nl/t/swtr/documents/wt2014_thingspeak.pdf.

[36] D. Parida, A. Behera, J. K. Naik, S. Pattanaik, and R. S. Nanda, “Real-time Environment Monitoring System using ESP8266 and ThingSpeak on Internet of Things Platform,” in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, May 2019, pp. 225–229, doi: 10.1109/ICCS45141.2019.9065451.

[37] I. Ozcelik and R. R. Brooks, *Distributed Denial of Service Attacks*. CRC Press, 2020.

[38] K. Sonar and H. Upadhyay, “A survey: DDOS Attack on Internet of Things,” *Int. J. Eng. Res. Dev.*, vol. 10, no. 11, pp. 58–63, 2014.

[39] F. Moldovan, P. Satmarean, and C. Oprisa, “An Analysis of HTTP Attacks on Home IoT Devices,” in *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, May 2020, pp. 1–6, doi: 10.1109/AQTR49680.2020.9129980.

[40] N. Abughazaleh, R. Bin, M. Btish, and H. M., “DoS Attacks in IoT Systems and Proposed Solutions,” *Int. J. Comput. Appl.*, vol. 176, no. 33, pp. 16–19, Jun. 2020, doi: 10.5120/ijca2020920397.

[41] K. Geetha and N. Sreenath, “SYN flooding attack - Identification and analysis,” in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Feb. 2014, pp. 1–7, doi: 10.1109/ICICES.2014.7033828.

[42] Q. A. Al-Haija, E. Saleh, and M. Alnabhan, “Detecting Port Scan Attacks Using Logistic Regression,” in *2021 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, Dec. 2021, pp. 1–5, doi: 10.1109/ISAECT53699.2021.9668562.

[43] L. Huraj, M. Simon, and T. Horak, “IoT Measuring of UDP-Based Distributed Reflective DoS Attack,” in *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, Sep. 2018, pp. 000209–000214, doi: 10.1109/SISY.2018.8524703.