IoT Botnet Detection and Classification using Machine Learning Algorithms for Improved Cybersecurity

Pham Van Quan[1[0009-0003-1625-0848]], Ngo Van Uc[2[0009-0005-0954-5618]], Do Phuc Hao[3 [0000-0003-0645-0021]], and Nguyen Nang Hung Van[4[0000-0002-9963-7006]]

[1, 2] Dong A University, Da Nang, Vietnam
quan10.work@gmail.com, ngovanuc.1508@gmail.com
[3] The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, Saint-Petersburg, Russian Federation
haodp.sut@gmail.com
[4] Danang University of Science and Technology, Da Nang, Vietnam
nguyenvan@dut.udn.vn

Abstract. In this era of advanced technology, the interconnection of various devices has become a crucial aspect. As a solution to optimize this interconnectivity, the Internet of Things has emerged as a prominent solution. However, ensuring the security of Internet of Things systems is equally essential. Timely detection of potential attacks on Internet of Things systems can effectively mitigate risks and minimize damages. In the present research, we investigated the efficacy of four algorithms, namely Decision Tree, K Nearest Neighbors , Random Forest , and Extreme Gradient Boosting , in detecting and classifying Internet of Things botnets. Our findings demonstrate that all four algorithms exhibit remarkable effectiveness in detecting and classifying botnets. Among these algorithms, the Extreme Gradient Boosting algorithm achieved the highest accuracy, while the Decision Tree algorithm exhibited the shortest execution time. This study highlights the potential of machine learning algorithms in detecting and reducing security threats in Internet of Things devices. By leveraging these algorithms, it is possible to detect and classify botnets promptly, thus minimizing the risk of security breaches in Internet of Things systems.

Keywords: Internet of Things, Supervised learning, Intrucsion detection, IoT Botnet, Cybersecurity.

1 Introduction

The concept of the Internet of Things (IoT) was initially proposed by K.Ashton (1999) as a solution to the increasing number of devices that require internet connectivity. The European Union Agency for Network and Information Security defines IoT as a cyber-physical ecosystem that comprises interconnected sensors and actuators that facilitate decision-making processes. Notably, IoT is a vital technology in Industry 4.0 [1]. Industrial IoT (IIoT) represents a specialized implementation of IoT that connects engines and industrial components to enhance the productivity and performance of industrial

activities. IIoT achieves this goal by providing real-time monitoring, efficient management, and control of industrial processes, assets, and operational time. Moreover, IIoT aims to reduce operational costs while improving overall efficiency [2].

However, developing IoT systems presents significant security challenges. IoT devices often have limited security and resource management capabilities. This is also an opportunity for Botnets to attack and infiltrate. When Botnets infiltrate, it can take control, get information and cause damage on many surrounding devices. Therefore, besides developing IoT systems, effective security measures are also needed. to protect against cyber threats and data breaches [3].

Preventing various attacks and intrusions and protecting data is very essential. Nowadays, There are a lot of intrusion detection systems(IDS) have been developed to respond to this problem. IDS has a mission to detect and report intrusion systems. In addition, IDS prevents malicious attacks and maintains performance during attacks. IDS is an important component of modern security systems[4].

Machine learning has introduced a fresh perspective to the development of IDSs. The present study focuses on the classification of nine malware captures of the IoT-23 dataset in both binary and multiclass scenarios. The models used in this study include Decision Tree (DT), K-Nearest Neighbor (KNN), Random Forest (RF), and XGBoost (XGB), which have demonstrated significant proficiency in classification tasks. The outcomes of this analysis yield promising results, thereby rendering it a suitable approach for intrusion detection. The research is structured into five distinct sections. Section 2 reviews previous works on machine learning and deep learning techniques that have been employed for intrusion detection. Section 3 elaborates on the dataset and models used, encompassing data preprocessing steps and evaluation metrics. Section 4 presents a comprehensive analysis of the findings. Finally, section 5 draws conclusions based on the research outcomes.

2 Related work

In recent years, research on IoT intrusion detection has been more attention. Research on this topic is becoming more necessary. Based on the results from previous research, there are many solutions to develop IoT intrusion detection systems. Understanding the strengths as well as the problems of previous research helps researchers find research directions that bring great benefits for building IDS systems. Therefore, previous studies are fundamental in providing knowledge for future development.

J. Hajji et al. [5] conducted a study to apply unsupervised machine learning algorithms, including K-means, PCA, and Autoencoder, to detect anomalous network traffic. A. Rahim et al. [6] employed statistical analysis to determine the most significant features and then applied several machine learning algorithms, such as DT, RF, and Naive Bayes (NB), to classify normal and malicious traffic. S. M. Z. Islam et al. [7] devised averaging and stacking models based on Support Vector Machine (SVM), RF, and Gradient Boosting Algorithms. Y. Li et al. [8] developed a bagging model from four machine learning models based on DT, KNN, Logistic Regression (LR), and RF

to classify network traffic as normal or malicious. P. H. Do [9] proposed a feature extraction method by dividing feature sets into various classes, followed by the application of machine learning algorithms to those attribute classes. These studies utilized a range of algorithms, including some combinations, to achieve high accuracy in their classification tasks.

Several researchers have explored the potential of deep learning models to detect anomalous behavior in network traffic. Alotaibi et al. [10] developed a Convolutional Neural Network (CNN) based model to classify network traffic as normal or malicious. Li et al. [11] utilized Long Short Term Memory (LSTM) and a Dense Neural Network to classify network traffic as normal or malicious. Similarly, Abdallah et al. [12] employed a deep learning model based on CNNs and LSTM. Kiani et al. [13] proposed a Deep Autoencoder Neural Network to learn the normal behavior of IoT network traffic and used it to detect anomalous behavior in real-time. Additionally, Rasool et al. [14] experimented with transfer learning models such as VGG16, ResNet50, and InceptionV3. These studies suggest the potential of deep learning models in detecting anomalies in network traffic.

Each of these researches presents different methods for detecting and classifying cyberattacks and all have shown good results. However, some researches have not mentioned the performance of the model as well as the execution time of the model, a few others only present detection or classification.

3 Method

This section will commence with a presentation of the dataset that will be used in the subsequent analysis. Preprocessing techniques will then be applied to ensure the accuracy and reliability of the data. Following that, we will delve into the process of data selection, visualization, formatting, and splitting. Our analysis will culminate with an evaluation of the effectiveness of the algorithms employed, accompanied by a comparative analysis of the outcomes.

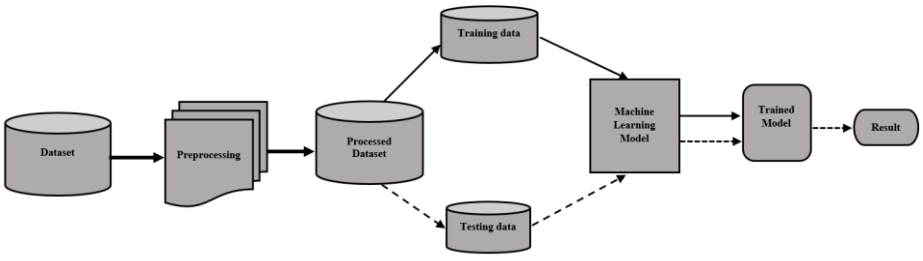

Fig. 1. Proposed method

3.1 Dataset

The IoT-23 dataset was entirely developed by the Stratosphere Laboratory in the Czech Republic and was published in 2020. This was a publicly available dataset that aims to

provide researchers with a benchmark for examining the security and privacy of Internet of Things (IoT) devices. The dataset contains captures of network traffic from 23 different types of IoT devices, such as smart plugs, cameras, smart locks, and smart thermostats, to name a few. The traffic captures were taken in a controlled environment where each device was connected to a separate Wi-Fi network and was subjected to various attack scenarios, including brute-force attacks, Mirai botnet attacks, injection attacks, etc. Metadata for each capture is included in the dataset, including the device type, firmware version, and attack type. The traffic distribution is shown in Table 1. The list with the description of all featuresis shown in Table 2.

Table 1. The traffic content of the IoT-23 dataset by executed attacks.

Attack Name	Flows
Part-Of-A-Horizontal-PortScan	213,852,924
Okiru	47,381,241
Okiru-Attack	13,609,479
DDoS	19,538,713
C&C-Heart Beat	33,673
C&C	21,995
Attack	9398
C&C-	888
C&C-Heart Beat Attack	883
C&C-File download	53
C&C-Torii	30
File download	18
C&C-Heart Beat File Download	11
Part-Of-A-Horizontal-PortScan Attack	5
C&C-Mirai	2

Table 2. IoT-23 dataset features.

Feature Name	Description
fields-ts	Start Time flow
uid	Unique ID
id.orig-h	Source IP address
id.orig-p	Source port
id.resp-h	Destination IP address
id.resp-p	Destination port
proto	Protocol
service	Type of Service (http, dns, etc.)
duration	Flow total duration
orig-bytes	Source—destination transaction bytes
resp-bytes	Destination—source transaction bytes
conn-state	Connection state
local-orig	Source local address
local-resp	Destination local address

resp-pkts	Destination packets
orig-ip-bytes	Flow of source bytes
history orig-pkts	History of source packets
missed-bytes	Missing bytes during transaction
tunnel-parents	Traffic tunnel
resp-ip-bytes	Flow of destination bytes
label	Name of type attack

3.2 Data Preprocessing

One of the most important parts of data preparation for model training is data preprocessing, which is considered the most time-consuming process. The data preprocessing is shown in Figure 2:

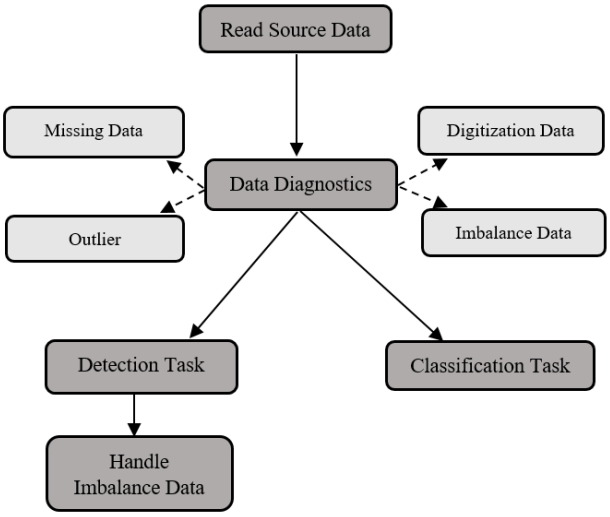

Fig. 2. The steps of data preprocessing

Data preprocessing starts with reading the data source. Next, we take care of the data quality by checking and diagnosing the data. In this step, we will solve some problems if any such as: missing data, data containing outliers, and data that must be numeric and imbalanced data. We look at each issue and come up with a suitable solution and avoid affecting the entire data.

For different tasks, we have to have different ways of handling data. Mainly here is the data imbalance at the layers due to the large number of different records. For the detection task, the difference between the malicious and benign classes is too large, so we perform sampling for the malicious group. For the classification task, we will merge several classes with a few records together to reduce data imbalance.

3.3 Analysis Method

Decision Tree

The decision tree algorithm is a widely used and popular type of supervised learning model that is effective in solving both classification and regression problems. Its structure consists of nodes that represent variables and branches that represent the relationship between these variables. At the root node, the algorithm considers the entire dataset, and through a series of binary decisions based on the input variables, it recursively partitions the data into smaller subsets. These partitions form internal nodes in the tree, and the leaves correspond to the predicted value of the target variable for each subset.

To build a decision tree, the algorithm follows a top-down approach that selects the best attribute to split the data based on a criterion such as information gain, Gini index, or entropy. Once the data is split, the algorithm recursively applies the same process to each of the resulting subsets until a stopping criterion is met, such as a predefined tree depth, minimum number of instances per leaf, or no further improvement in the predictive performance.

To update and calculate node values in a decision tree, two formulas are used: the impurity measure and the criterion for selecting the best split. The impurity measure quantifies the degree of homogeneity or heterogeneity of the target variable within a node, while the criterion for selecting the best split determines which attribute provides the highest information gain or the lowest impurity after splitting the node.

The entropy of probability distribution $\boldsymbol{p} = (p_1, p_2, p_3, \dots, p_n)$ satisfied $\sum_{i=1}^{n} p_1 = 1$

$$H(p) = -\sum_{i=1}^{n} p_i \times \log p_i \tag{1}$$

The information gain is a node test formula that yields the amount of that node information that remains after switching to $\boldsymbol{k}$ child node.

$$Gain(p) = H(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} \times H(i)\right) \tag{2}$$

Random Forest (RF)

The Random Forest algorithm is a well-known and widely used supervised classification technique, that leverages the power of multiple decision trees to improve its predictive accuracy. Specifically, the algorithm generates multiple decision trees and aggregates their results to obtain the final prediction. The number of decision trees in the forest is an important parameter that influences the accuracy of the algorithm. Interestingly, as the number of decision trees increases, the accuracy of the algorithm also tends to increase, although there is a point beyond which further increasing the number of trees has a negligible effect on accuracy. One of the main advantages of Random Forest is its ability to handle high-dimensional data, while requiring less computational resources compared to other competing algorithms. This is mainly due to the fact that the

algorithm samples a subset of the features at each node of the decision tree, which reduces the dimensionality of the problem and leads to faster convergence.

K-Nearest Neighbor (KNN)

The K-Nearest Neighbor (KNN) algorithm is a prevalent supervised learning approach used in the domains of data mining and machine learning. KNN is categorized as a "lazy learning" algorithm, implying that it does not acquire knowledge from the training data. Instead, computations are performed only when it needs to predict the labels of new data. To predict the label of a new data point, KNN uses the mean of the k-nearest labels in its vicinity.

$$label\ of\ new\ data = \frac{\sum_{i=1}^{k} label(i)}{k} \quad (3)$$

Extreme Gradient Boosting (XGBoost)

XGBoost, or Extreme Gradient Boosting, is a cutting-edge algorithm for tackling supervised mathematical problems with a remarkable level of precision, rivalling even the performance of deep learning models. It is capable of processing tabular data of varying size and structure, including classification data. The version of XGBoost employed in the present study is notable for its rapid training rate, which surpasses that of many other algorithms.

3.4 Performance Evaluation

In assessing the algorithmic models described earlier, diverse techniques were employed to gauge the precision of the outcomes and derive comprehensive findings for each model. This study involved several fundamental concepts, such as TP, TN, FP, and FN. TP signifies the count of true positives that have been accurately determined, while TN is the number of true negatives that have been correctly identified. FP represents the actual number of positive cases that have been erroneously classified as negative, whereas FN denotes the count of negative cases that have been mistakenly classified as positive.

Precision (PRE)

Precision is a performance metric utilized to assess a model's effectiveness by determining the proportion of accurately identified positive instances. This measure can be mathematically calculated through the use of a formula, which takes into account the number of true positives and false positives. Specifically, precision is computed by dividing the number of true positives by the sum of true positives and false positives.

$$PRE = TP/(TP + FP) \quad (4)$$

Accuracy (ACC)

Accuracy is a performance metric utilized to gauge a model's efficacy by determining the proportion of correct predictions out of the total number of predictions. This measure can be mathematically calculated through the use of a formula, which considers the number of true positives and true negatives. Specifically, accuracy is computed by dividing the sum of true positives and true negatives by the total number of predictions.

$$ACC = (\mathrm{TP} + \mathrm{TN}) \,/\, (\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}) \quad (5)$$

Recall Score (RE)

The recall score is a performance metric utilized to evaluate a model's efficacy in correctly identifying actual positive instances. This measure can be mathematically calculated through the use of a formula, which incorporates the number of true positives and false negatives. Specifically, the recall score is computed by dividing the number of true positives by the sum of true positives and false negatives.

$$RE = TP/(TP + FN) \quad (6)$$

F1 Score for Binary class

The F1 score is a performance metric that is obtained by averaging both the accuracy and recall scores. This measure is widely used in assessing the overall effectiveness of a model since it provides a comprehensive view of false positives and false negatives. Specifically, the F1 score is calculated as the harmonic mean of precision and recall. The formula for computing the F1 score takes into account both the number of true positives, false positives, and false negatives, thereby providing a more balanced evaluation of a model's performance.

$$F1 - score = 2 \times \frac{PRE \times RE}{PRE + RE} \quad (7)$$

F1 Score for Multi-class

For multi-class classifiers, there are two options: micro-averaging (taking into account the frequency of each class) and macro-averaging (all classes are counted equally). Macro-averaging is more suitable for data sets that are relatively uniform in size, while micro-averaging is more suitable for data sets has a large disproportion between the sizes of the classes[16]. The formulas for them are:

$$F1 - micro = \sum_{i=1}^{n} [f1(i) \times \frac{supportscore(i)}{setsize}] \quad (8)$$

$$F1 - macro = \sum_{i=1}^{n} [F1(i) \times \frac{1}{n}] \quad (9)$$

4 Results and Discussion

4.1 Results

Binary Classification

Upon conducting model training and testing, diverse outcomes were obtained. Notably, there was little disparity among the various results. A comprehensive overview of the accuracy and training duration of each model is depicted in the Table 3:

Table 3: Some metrics of binary classification

Evaluation	DT	RF	KNN	XGB
ACC	99.9%	89.5%	99.6%	99.8%
PRE	100.0%	88.0%	100.0%	100%
RE	100.0%	86.0%	99.0%	100%
F1	100.0%	87.0%	100.0%	100%
Time (s)	5.9	40.2	58.2	99.4

In the context of binary classification, DT stands out as the most effective model with the shortest processing time. This algorithm is particularly adept at handling datasets characterized by high error rates, thanks to its binary "yes" or "no" decision-making mechanism. Conversely, KNN relies heavily on data attributes, which can result in significant errors when dealing with dissimilarities between classes that lead to mean values. While XGB is a potent algorithm that yields promising outcomes, its relatively lengthy processing time makes it impractical for real-world applications.

Multi-classificaion

The findings obtained from subjecting the models to multilayer attacks closely align with those reported earlier. Specifically, Random Forest's outcomes were comparatively inferior to those of other models. Additionally, training times for models utilized in multithreaded attacks were notably extended. For further clarification, a detailed breakdown of the outcomes and training duration for each model is presented in the Table 4:

Table 4: Some metrics of multi-classification

Evaluation	DT	RF	KNN	XGB
ACC	99.9%	78.2%	99.8%	99.9%
PRE	99.9%	59.6%	99.7%	100.0%
RE	99.7%	46.1%	98.7%	99.8%
F1	99.6%	49.8%	99.2%	99.9%
Time (s)	11.5	206.1	85.2	956.2

Models evaluation

In order to ascertain the suitability of a given model for a particular problem, performance evaluation techniques are utilized. A range of methodologies may be employed, including but not limited to accuracy, precision, recall, and F1-score measurements. These methods serve as reliable indicators of a model's ability to effectively address the original problem at hand.

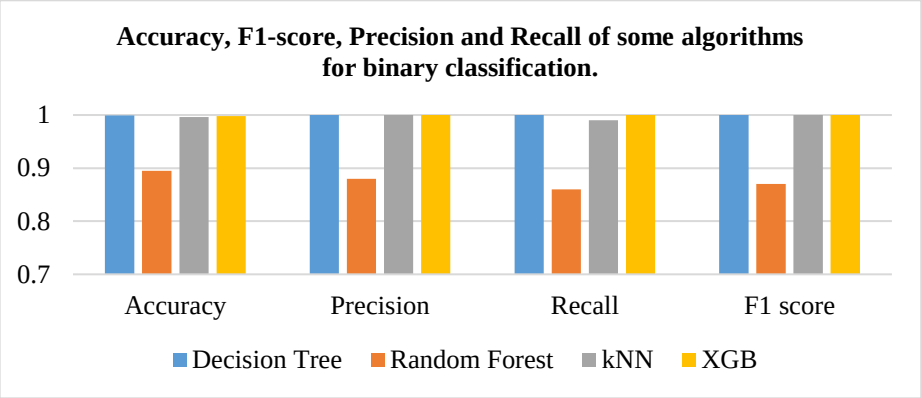

Fig. 3. Accuracy, F1-score, Precision and Recall of some algorithms for binary classification.

Figure 3 presents an overview of the accuracy, precision, recall, and F1-score metrics associated with the DT, RF, KNN, and XGB algorithms, as applied to binary classification. As per the results, it is evident that the DT and XGB algorithms have performed exceptionally well, surpassing their counterparts. Moreover, in terms of training time, the Decision Tree algorithm has proven to be more efficient, thereby achieving superior performance overall.

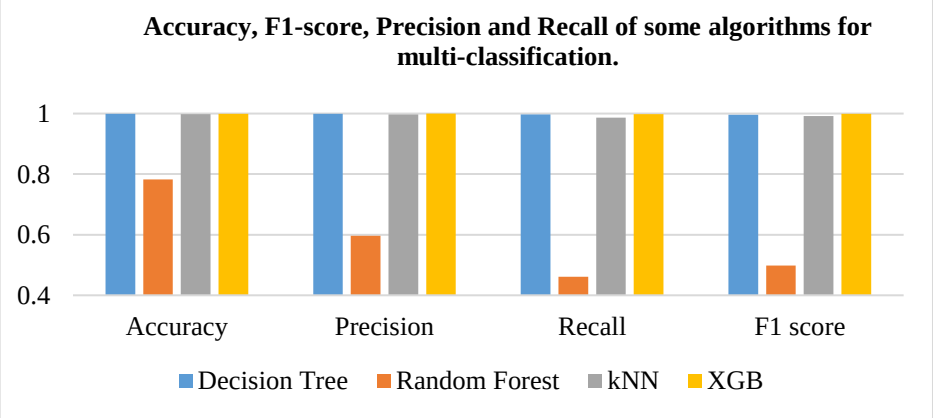

Fig. 4. Accuracy, F1-score, Precision and Recall of some algorithms for multi-classification.

The performance of four algorithms, namely Decision Tree, Random Forest, KNN, and XGB, was evaluated for the multiclass problem in figure 4. The evaluation was based on the accuracy, precision, recall, and F1-score metrics. The results indicate that the Decision Tree algorithm outperforms the other algorithms in terms of these performance metrics.

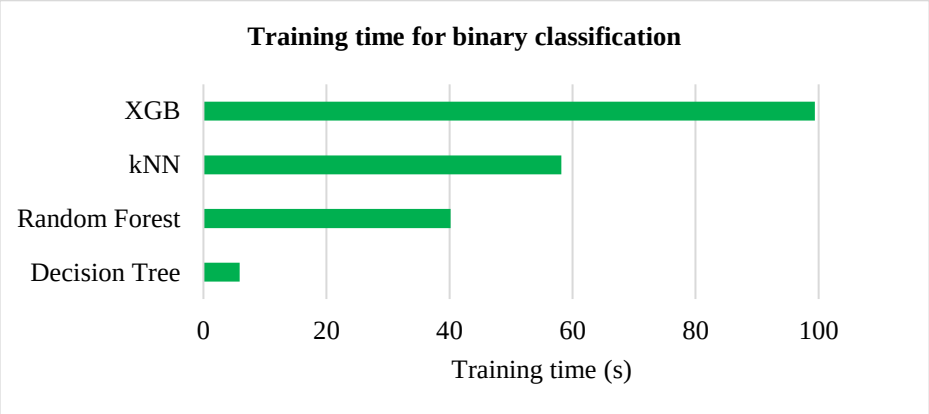

Fig. 5. Training time for binary classification

The training time of algorithms for the binary classification problem is presented in Figure 5. According to the results, the XGB algorithm takes significantly longer time to train compared to the other algorithms. On the other hand, the decision tree algorithm has shown the best training time among all the algorithms.

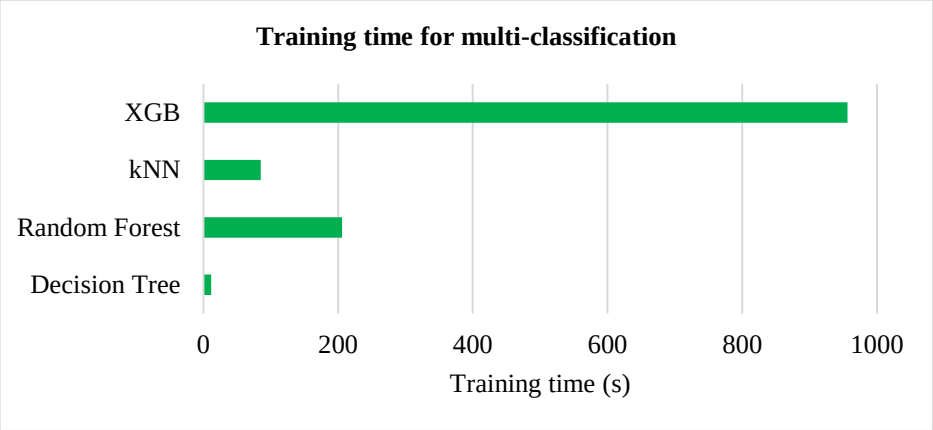

Fig. 6. Training time for multi-classification

In the multi-classification problem, the training time of the algorithms is presented in figure 6. The results illustrate that the XGB algorithm takes significantly longer to train compared to the other algorithms, with a training time almost 100 times that of the

decision tree algorithm. On the other hand, the decision tree algorithm exhibits the best performance in terms of training time.

4.2 Discussion

In this paper, we tested the impact of four algorithms on the IoT-23 dataset in recognition and classification. Models work well on separate tasks. However, in many tasks, the models do not perform well and are heavily biased.

In the future, we plan to experiment with some other machine learning algorithms and artificial neural networks. In addition, we will work on several other IoT datasets and extend them. For the actual implementation of the models, we plan to test the best features that improve speed without sacrificing performance too much.

5 Conclusions

The emergence of machine learning models for IoT intrusion detection is a relatively recent development, but it has already yielded significant results. In this study, several machine learning algorithms, including Decision Tree, Random Forest, KNN, and XGB, were applied to the Iot-23 dataset to evaluate their efficacy in identifying nine different malware bot captures. The results demonstrated that all four algorithms were effective for IoT intrusion detection on the IoT-23 dataset, although their performance may vary depending on the specific features of the dataset as well as the types of IoT devices analyzed.

These findings hold great promise for improving IoT security networks through the application of machine learning algorithms. Nonetheless, more research is needed to explore the performance of other algorithms and to develop even more accurate and effective intrusion detection systems for IoT networks.

References

1. Williams, P., Dutta, I.K., Daoud,. H, & Bayoumi, M.: A survey on security in internet of things with a focus on the impact of emerging technologies. Elsevier. (2022).
2. Khan, W.Z., Rehman, M.H., Zangoti, H.M., Afzal, M.K., Armi, N., & Salah, K.: Industrial internet of things: Recent advances, enabling technologies and open challenges. Elsevier. (2020).
3. Vitorino, J., Andrade, R., Praça, I., Sousa, O., & Maia, E.: A Comparative Analysis of Machine Learning Techniques for IoT Intrusion Detection. Foundations and Practice of Security (pp.191-207).Springer.(2022).
4. Haq, N.F., Onik, A.R., Hridoy, M.A.K, Rafni, M., Shah, F.M., & Farid, D.M.: Application of Machine Learning Approaches inIntrusion Detection System: A Survey. Article Published in International Journal of Advanced Research in Artificial Intelligence(IJARAI), Volume 4 Issue 3. (2015).
5. Hajji, J., Khalily, M., Moustafa, N., & Nelms, T. IoT-23: A Dataset for IoT Network Traffic Analysis. Springer. (2019).

6. Rahim, A., Razzaque, M.A., Hasan, R., & Hossain, M.F. Effective IoT Network Security through Feature Selection and Machine Learning Techniques. IEEE. (2020).
7. Islam, S.M.Z, Bhuiyan, M.Z.H, & Hasan, R. Fusion of Machine Learning Models for Intrusion Detection in IoT Networks using the IoT-23 Dataset. IEEE. (2020)
8. Li, Y., Qiu, L., Chen, Y., & Chen, Y. Ensemble-based Intrusion Detection System for IoT Networks using the IoT-23 Dataset. IEEE. (2020)
9. P. H. Do, T. D. Dinh, D. T. Le, V. D. Pham, L. Myrova and R. Kirichek, "An Efficient Feature Extraction Method for Attack Classification in IoT Networks," 2021 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)
10. Alotaibi, F., Al-Qaness, M.A., Abunadi, A., & Alghazzawi, M.A. A Deep Learning Approach for Intrusion Detection in IoT Networks using the IoT-23 Dataset. IEEE. (2020).
11. Li, J., Hu, C., Yang, K., Zhang, X., & Lu, J. An IoT-23 based IoT Intrusion Detection System using Deep Learning. IEEE. (2020).
12. Abdallah, A., Khalil, I., Al-Emadi, N., Almohaimeed, A., & Kim, H. Real-Time IoT Botnet Detection Using Deep Learning on IoT-23 Dataset. IEEE. (2020)
13. Kiani, A.T., Abbas, R.A., Abbasi, A.Z., & Khan, M.K. Deep Learning-based Anomaly Detection for IoT Networks using the IoT-23 Dataset. IEEE. (2020)
14. Rasool, S., Saeed, S., Farooq, F., & Madani, A. A Comparative Study of Transfer Learning Approaches for IoT Malware Detection Using IoT-23 Dataset. IEEE. (2021).
15. Garcia, S., Parmisano, A., & Erquiaga, M.J. IoT-23: A labeled dataset with malicious and benign IoT network traffic. Stratosphere IPS. (2020).
16. Stoian, N.A. Machine Learning for Anomaly Detection in IoT Networks : Malware analysis on the IoT-23 data set. EEMCS: Electrical Engineering, Mathematics and Computer Science. (2020)