Decision Trees Versus Random Forest for DDoS Attack Classification*

Artur Kostera

Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, Poland

Abstract

Distributed Denial Service (DDoS) attack classification is essential to prevent network server or application overload. Massive network traffic can cause overload, which can end with shutdown. Consequently, detecting such attacks allows for the continuity of application services and their security. In this paper, decision trees and random forests to detect such attacks were proposed and evaluated as a classification task. For this purpose, a publicly available data set was used and the operation of both tools was verified to compare them. The experiments showed high-accuracy results for both tools in different numbers of samples in datasets.

Keywords

decision tree, random forest, comparison, DDoS, security

1. Introduction

Network security is related to many different issues [1, 2]. One of them is the problem of trying to overload various services or resources in the network. This is done by increasing the amount of network traffic to a specific resource simultaneously multiple times. The consequences of such attacks are the unavailability of services due to the attack carried out. Moreover, the next stage is a decrease in the trust of users or customers in creators or companies. In addition, even a single attack can be dangerous due to the possibility of data theft [3].

It is worth paying attention to the possibilities of protection against DDoS attacks [4, 5], which primarily involve monitoring network traffic. Such action allows for early detection of increasing network traffic and, therefore, also prevention. Content Delivery Network [6] is also used for the discussed purpose, or the dispersion of data centers. An interesting solution is traffic filtering, which is implemented by analyzing IP addresses.

Recent years have also shown the possibility of constructing protection and security systems against various attacks, not DDoS. It is worth emphasizing here the options of using blockchain-related techniques [7], which enable secure data storage. In addition, many different encryption techniques can help protect data against network attacks, including classic techniques such as RSA, AES, and DES [8, 9]. The presented approaches allow for the protection of data that may be one of the targets of network attacks. Protection systems should focus on securing data and analyzing the current situation in the network. An example of a solution based on security controllers in the software-defined network model is presented in [10]. The authors described a DDoS attack detection model using a machine learning model like XGBoost. An interesting framework that identifies information protection mechanisms by analyzing various regulations such as GDPR or NIS2 is described in [11]. An essential element is the analysis of existing limitations of resources such as energy or computing power. In [12], the researchers drew attention to the effects of these elements and described a framework party on feature engineering. Again, in [13], the integration of the DDoS detection module with the SDN-WISE controller was proposed. The indicated solutions show that network security is an important research goal.

This paper proposes using decision trees and random forests for DDoS classification. It aims to adapt and compare these tools to the attack detection capabilities. Detection analysis was carried out using different values of set partitions to define a more effective tool.

*IVUS 2025: Information Society and University Studies 2025, May 15, Kaunas, Lithuania

ak307879@student.polsl.pl (A. Kostera)

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-06-12

CEUR-WS.org/Vol-4213/paper01.pdf

2. DDoS

A DDoS attack is a cyber attack on a computer system or network service to prevent it from operating[14]. It uses multiple infected devices that simultaneously send requests or data to the victim. In this way, the attack disrupts the target's operation by flooding it with massive Internet traffic. The attack is usually carried out by computers over which control has been taken using special software. On a given signal, the computers start simultaneously attacking the victim's system, inundating it with false attempts to use the services it offers. For each such call, the attacked computer has to allocate certain resources such as memory, CPU time or network bandwidth, which, with many requests, leads to the exhaustion of available computer resources.

DDoS cyber attacks are divided into three types: Volumetric attacks, Attacks on network protocols, and Attacks on the application layer[15]. Volumetric attacks generate huge amounts of fake traffic that occupy available Internet connections. Such attacks operate at the network layer and can be based on sending UDP packets (so-called UDP flood) or amplifying traffic using open DNS servers (DNS amplification).

Attacks on network protocols involve sending invalid or redundant requests at the transport and network levels. For example, an attacker can send many connection-initiating requests (SYN packets) but not complete the process, which means not sending ACK packets. This approach is called a SYN flood. Another way to attack is to send huge ICMP packets (ping) that cause the system to crash (Ping from Death). Attacks on the application layer involve sending many HTTP requests or other requests directly to the application. An example of such an attack is HTTP flood, which is the repeated sending of HTTP GET or POST requests to overload the web server. Another such attack is Slowloris, which sends partial HTTP requests that force the server to keep multiple connections open.

Volumetric attacks attempt to block network bandwidth by sending up to hundreds of gigabits per second. Attacks on network protocols consume server resources such as memory or CPU by sending non-existent or faulty requests. Application layer attacks aim to disrupt specific applications, such as websites or databases, with many requests sent directly to the application. The last effect of the attack is more difficult to detect than the others because it can resemble normal user traffic. Although the operation of the DDoS above attacks varies, in general, the goal of these attacks is to disrupt the operation of a particular device.

3. Dataset description

The dataset used to validate selected tools is available on Kaggle [1]. It consists of 6321980 entries corresponding to ordinary Internet communications and 6472647 denoting various DDoS attacks, including but not limited to those described earlier. The database consists of 85 columns and 12794627 rows. Each column represents a characteristic of a specific Internet connection. A single row includes details such as the IP address that starts the connection, the exact date and time at which the connection began, the ports and protocols that are used, and all sorts of information about the amount of data transferred and the duration of the connection. The last column consists of two values, "DDoS" for the lines that describe a DDoS attack and "Benign" for the lines that describe a normal connection.

The first step was analyzing and preparing the dataset, removing some columns like: Flow ID, Src IP, Dst IP, Timestamp, Flow Byts/s, and Flow Pkts/s. This is because each column consists exclusively of or contains troublesome data types. The system only supports numeric values, so columns with IP addresses, dates, and values of type NaN and Infinite are removed. In addition, the last "Label" column containing data classes is moved to a separate variable so that the system does not consider it a feature.

[1] https://www.kaggle.com/datasets/devendra416/ddos-datasets

4. Decision Tree

A decision tree is a structure used in machine learning, statistics and data analysis to make decisions based on given inputs [16]. It is a predictive model that works by iteratively partitioning data based on logical conditions. Each step (node) in the tree corresponds to a test for the value of a particular feature, and each leaf represents an outcome (e.g., class assignment or predicted value).

A tree comprises a root, nodes, branches and leaves. The root is the first node of the tree, from which the entire decision-making process begins. It represents the entire data set. Nodes answer a question or decision rule based on a data feature. Branches are possible answers to a question at a node. Each branch leads to the next node or leaf. Leaves are the final results of the tree. Each leaf contains a class assignment (for classification) or a predicted value (for regression).

The classification process by decision tree starts from the root, where it is calculated how many entries from the entire database belong to each class. Then, using the following formula, the Gini coefficient is calculated, determining the impurity of the set and returning a value between 0 and 1. If all samples belong to one class, the coefficient will return a value of 0. If the samples are evenly divided between classes, it will return a value of 1. This coefficient is calculated using the following equation:

$$Gini(t) = 1 - \sum_{i=1}^{k} p_i^2, \tag{1}$$

where k is the number of classes and p_i is the proportion of samples at node t that belong to class i [17, 18].

The tree then divides the root to form two nodes; the division is relative to the feature. What trait will divide the split at a given stage is essential, so the tree temporarily divides the set into nodes against each trait to compare the effectiveness of each split and choose the best one. As for the root, a Gini coefficient is calculated for each of the splits. According to the formula below, the total coefficient of each division is calculated and then the one that achieves the smallest value is selected. The formula is described as:

$$Gini_{split} = \frac{n_{left}}{n_{total}} \cdot Gini(left) + \frac{n_{right}}{n_{total}} \cdot Gini(right), \tag{2}$$

where n_{left},n_{right} is the number of samples in the left and right nodes and n_{total} is the number of samples in the parent node.

Following the above-described rules, the tree divides the newly created two nodes. This process is repeated until one of the end conditions occurs. Final conditions determine whether a node will be further divided or considered a leaf. Examples of conditions are:

- No further improvement in the quality of the split - the tree will stop growing a node if it cannot find a split that leads to an improvement in the purity of the node.
- Minimum number of samples in a node - if a node consists of several samples less than the minimum, it is two samples for this model, and it will stop growing.
- Minimum number of samples in a leaf - if a split creates a leaf with fewer samples than the specified minimum of the leaf, for this model, if it is single sample, the split will not be made.
- Maximum tree depth - it is possible to specify after what depth the tree should stop growing, in the model such a number is not specified allowing any depth of the tree.
- Lack of diversity of classes in a node - this situation occurs when all samples in a node belong to one class.

5. Random Forest

Random Forest is a machine learning algorithm in the ensemble methods group. The ensemble methods group consists of machine learning models that combine several basic models to create a single optimal

predictive model. Random forest algorithms are built from multiple independent decision trees from different data samples. The algorithm's results are obtained by averaging the results of the trees (for regression) or by selecting a result consistent with the majority of the trees (for classification). This solution makes the algorithm, among other things, less prone to overfitting, has higher prediction accuracy, and is better at detecting high-importance features than a simple single decision tree [19].

Since random forests are built from decision trees, their main calculations are performed as described in the previous section. The data that each tree in the forest has access to distinguishes the operation of forests from trees. Let's assume that the entire database consists of 100 samples, in which case, each tree will get 100 samples for classification. Still, they will be randomly selected from the entire database with repetition, which is the bootstrap method. This means that some entries will not go to a given tree, and there is also an impossibly small chance that a tree will receive the same sample 100 times. Each tree independently draws its dataset [20].

Another difference is the division into nodes. When an individual tree creates new nodes, it cannot access all features. Each tree independently draws "m" features at each splitting, where $m=\sqrt{p}$ and p is the number of all features. This arrangement prevents the dominance of features that have more influence in the entire set, leading to more diverse trees, and reduces the correlation between individual trees, which increases the efficiency of averaging the results[21]. Since "m" is selected anew when each split is created, a feature already used in that tree to divide may end up in the "m" set again. When each tree of the forest stops expanding with new nodes, the model can begin classification. The sample to be classified goes to each tree and the forest itself returns a score based on the response of each of its trees. The forest counts the decisions of the trees and then returns a score according to the following formula:

$$\hat{y} = argmax_k \left(\sum_{i=1}^{T} 1\left(\hat{y}_i = k\right) \right), \tag{3}$$

where, $1()$ is an indicator function that equals 1 if tree T_i predicts class k_i and 0 otherwise. The final prediction $\hat{y}$ is the class that receives the most votes from all trees.

6. Experiments

The model performance is evaluated using **accuracy**, **precision**, **recall**, **F1-score**, and **support** metrics. Most of these metrics (except support) range from 0 to 1, where 1 indicates the best performance.

In detail, the used metrics are defined as follows:

- **Accuracy** is defined as the ratio of correctly classified samples to the total number of samples:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4}$$

  where TP denotes True Positives (correctly predicted positive cases), TN denotes True Negatives (correctly predicted negative cases), FP denotes False Positives (negative cases incorrectly predicted as positive), and FN denotes False Negatives (positive cases incorrectly predicted as negative).
- **Precision** indicates what proportion of samples classified as a given class actually belong to that class:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

  High precision means that few samples have been misclassified as belonging to the class.
- **Recall** measures how well the model detects all samples belonging to a given class:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

  It represents the percentage of correctly classified samples relative to all samples belonging to that class.

- **F1-score** is the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

  It is used as a balanced measure of model performance, especially when it is essential to minimize both False Positives and False Negatives simultaneously.
- **Support** refers to the number of samples belonging to each class in the test data. It is not a measure of the model's performance but provides information about the size of each class, which can affect the interpretation of other metrics.

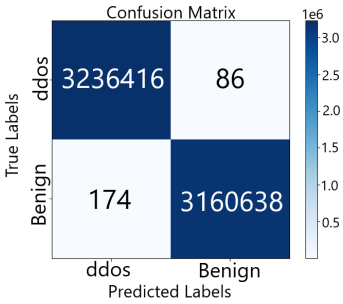
In addition to the above metrics, a confusion matrix determines the model's effectiveness. The confusion matrix is a table that shows the results of the classification performed by the model in detail. It divides correct and incorrect classifications into categories, which allows us to understand precisely how the model got it wrong and in which cases it was effective. The matrix has rows and columns where the rows represent the actual classes in the data and the columns represent the predicted classes by the model. The table is presented graphically, and the intensity of the blue color is used to determine the number of samples belonging to a given category; the darker the blue color, the more samples belong to a given table cell. In addition to the color, a number determines the same thing, but only in a more unambiguous way.

The testing involved comparing the model's effectiveness for different base divisions in a training and test set. Starting with a split of 50% by 50%, each subsequent test increased the training group by 10%, decreasing the test group until the split was 90% by 10%. Such a comparison was carried out for both the tree and forest models.

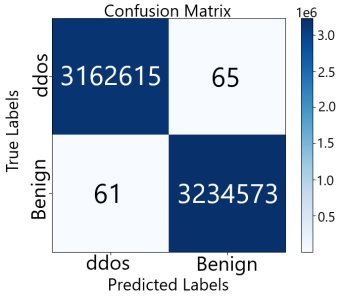
The first test was to split the set 50:50. The results are presented in Tab. 1 and on the confusion matrices in Fig. 1a and 1b. As we can see from the confusion matrix for decision trees, even though the metrics reached almost 100%, the model made 260 mistakes. The metrics achieved total efficiency because the number of errors the model made is imperceptible. We can notice in the picture that the model was wrong more often in recognizing Benign samples as DDoS than vice versa. In the case of using random forests, the results also indicate very good efficiency. However, based on metrics, a much more accurate classification can be observed in the case of random forest, where in both classes, more samples were classified correctly.

The next test was to increase the training set to 60% of all data, the results of which are presented in Fig. 1c and 1d and in Tab. 2. Both classifiers performed better, which is the effect of increasing the number of samples in the training set. However, the classification results for decision trees indicate a larger decrease in misclassified values. Misclassification of DDoS attacks with an increase in the number of samples in the training set contributed to a decrease of only five samples, which indicates an improvement but not a significant one. The subsequent increase in the training set was 70%, and 30% in the test set. The obtained results are presented in Fig. 1g and 1h and in Tab. 3. The metric values have slightly increased, but the confusion matrix analysis indicates a much better adaptation of random forests to detect DDoS attacks. The number of incorrectly classified attacks was only 27 samples. Compared to decision trees, this is a difference of 17 samples indicating an attack and a difference of 63 samples that are not an attack but are classified as an attack.

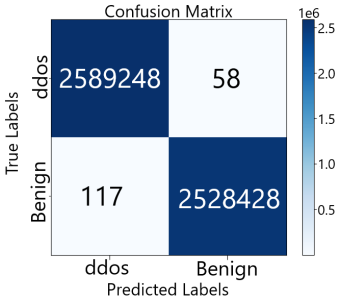
The next test was performed with a split of 80% of all samples in the training set and the rest in the test set (see Fig. 1i and 1j and in Tab. 4). The results show an improvement in the classification results for decision trees, but random forest is still more effective in this respect. However, increasing the training data slightly improved the classification result of samples labeled as DDoS because it is 1 sample. The last test with increasing the number in the training set showed very good results, but it should be noted that the test set is much smaller (see Fig. 1e and 1f and in Tab. 5). The tests indicate a better adaptation of random forest to DDoS data analysis than decision trees.



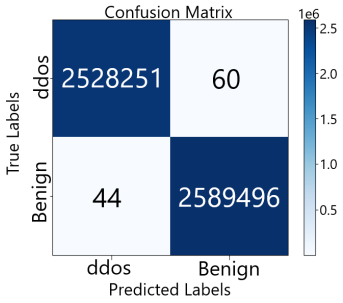
(a) Decision Trees (50:50)



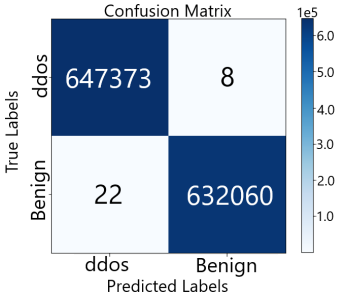
(b) Random Forests (50:50)



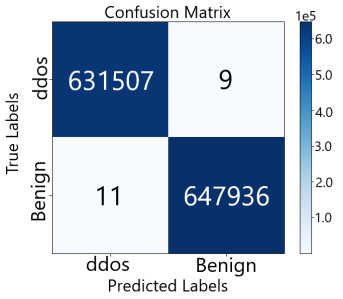
(c) Decision Trees (60:40)



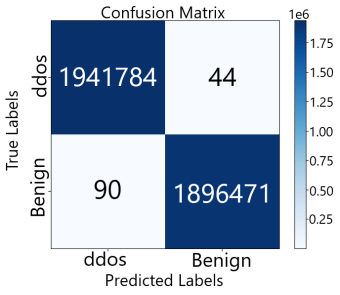
(d) Random Forests (60:40)



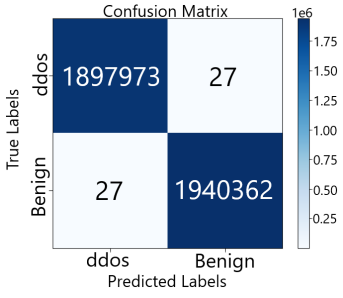
(e) Decision Trees (90:10)



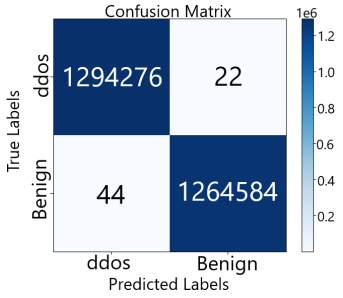
(f) Random Forests (90:10)



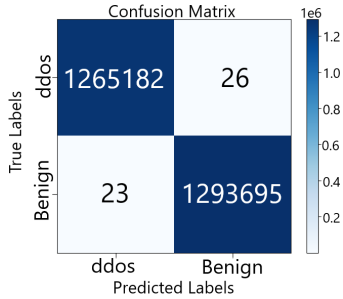
(g) Decision Trees (70:30)



(h) Random Forests (70:30)



(i) Decision Trees (80:20)



(j) Random Forests (80:20)

Figure 1: Comparison of confusion matrices for decision trees and random forests for various training-validation splits.

Table 1
Comparison of results between the 50%-50% training-to-validation split.

Metric	Decision Tree	Random Forest
Support (Benign)	3,160,812	3,162,680
Support (DDoS)	3,236,502	3,234,634
Accuracy	99.9959%	99.9980%
Precision (DDoS)	99.9946%	99.9981%
Recall (DDoS)	99.9973%	99.9979%
F1-score (DDoS)	99.9960%	99.9980%
Precision (Benign)	99.9973%	99.9980%
Recall (Benign)	99.9945%	99.9981%
F1-score (Benign)	99.9959%	99.9981%

Table 2
Comparison of results between the 60%-40% training-to-validation split.

Metrics	Decision Tree	Random Forest
Support (Benign)	2,528,545	2,528,311
Support (DDoS)	2,589,306	2,589,540
Accuracy	99.9966%	99.9980%
Precision (DDoS)	99.9955%	99.9983%
Recall (DDoS)	99.9978%	99.9976%
F1-score (DDoS)	99.9966%	99.9980%
Precision (Benign)	99.9977%	99.9977%
Recall (Benign)	99.9954%	99.9983%
F1-score (Benign)	99.9965%	99.9980%

Table 3
Comparison of results between the 70%-30% training-to-validation split.

Metrics	Decision Tree	Random Forest
Support (Benign)	1,896,561	1,898,000
Support (DDoS)	1,941,828	1,940,389
Accuracy	99.9965%	99.9986%
Precision (DDoS)	99.9954%	99.9986%
Recall (DDoS)	99.9977%	99.9986%
F1-score (DDoS)	99.9965%	99.9986%
Precision (Benign)	99.9977%	99.9986%
Recall (Benign)	99.9953%	99.9986%
F1-score (Benign)	99.9965%	99.9986%

7. Conclusions

This paper presents DDoS attacks that can paralyze certain services or resources and even contribute to data theft. Two classic tools were used to detect DDoS attacks, such as decision trees and random forests. An analysis of a publicly available data set was presented and the influence of the amount of training data needed to obtain good classification results was shown. The presented results indicated the effectiveness of both tools, as shown by the determined metrics. However, random forests turned out to be more accurate classifiers in each case, which indicates their better adaptation to such data.

Table 4
Comparison of results between the 80%-20% training-to-validation split.

Metrics	Decision Tree	Random Forest
Support (Benign)	1,264,628	1,265,208
Support (DDoS)	1,294,298	1,293,718
Accuracy	99.9974%	99.9981%
Precision (DDoS)	99.9966%	99.9982%
Recall (DDoS)	99.9983%	99.9979%
F1-score (DDoS)	99.9975%	99.9981%
Precision (Benign)	99.9983%	99.9980%
Recall (Benign)	99.9965%	99.9982%
F1-score (Benign)	99.9974%	99.9981%

Table 5
Comparison of results between the 90%-10% training-to-validation split.

Metrics	Decision Tree	Random Forest
Support (Benign)	632,082	631,516
Support (DDoS)	647,381	647,947
Accuracy	99.9977%	99.9984%
Precision (DDoS)	99.9966%	99.9983%
Recall (DDoS)	99.9988%	99.9986%
F1-score (DDoS)	99.9977%	99.9984%
Precision (Benign)	99.9987%	99.9986%
Recall (Benign)	99.9965%	99.9983%
F1-score (Benign)	99.9976%	99.9985%

Declaration on Generative AI

During the preparation of this work, the author used OpenAI ChatGPT-4 in order to: suggest phrasing improvements and provide grammar and spelling checks. After using this tool, the author reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Lagraa, M. Husák, H. Seba, S. Vuppala, R. State, M. Ouedraogo, A review on graph-based approaches for network security monitoring and botnet detection, International Journal of Information Security 23 (2024) 119–140. doi:10.1007/s10207-023-00742-7.

[2] Q. Chen, D. Li, L. Wang, Blockchain technology for enhancing network security, Journal of Industrial Engineering and Applied Science 2 (2024) 22–28. doi:10.5281/zenodo.12786722.

[3] M. Olabim, A. Greenfield, A. Barlow, A differential privacy-based approach for mitigating data theft in ransomware attacks, Authorea Preprints (2024). doi:10.22541/au.172625434.48862692/v1.

[4] A. Jaszcz, D. Połap, Aimm: Artificial intelligence merged methods for flood ddos attacks detection, Journal of King Saud University-Computer and Information Sciences 34 (2022) 8090–8101. doi:10.1016/j.jksuci.2022.07.021.

[5] M. H. Ali, M. M. Jaber, S. K. Abd, A. Rehman, M. J. Awan, R. Damaševičius, S. A. Bahaj, Threat analysis and distributed denial of service (ddos) attack recognition in the internet of things (iot), Electronics 11 (2022) 494. doi:10.3390/electronics11030494.

[6] M. Imthiyas, S. Wani, R. A. A. Abdulghafor, A. A. Ibrahim, A. H. Mohammad, Ddos mitigation: A review of content delivery network and its ddos defence techniques, International Journal

on Perceptive and Cognitive Computing 6 (2020) 67–76. DOI not available; article accessible at https://journals.iium.edu.my/kict/index.php/IJPCC/article/view/160.

[7] R. Chaganti, B. Bhushan, V. Ravi, A survey on blockchain solutions in ddos attacks mitigation: Techniques, open challenges and future directions, Computer Communications 197 (2023) 96–112. doi:10.1016/j.comcom.2022.10.026.

[8] S.-Q. Yeow, K.-W. Ng, Neural network based data encryption: A comparison study among des, aes, and he techniques, JOIV: International Journal on Informatics Visualization 7 (2023) 2086–2094. doi:10.30630/joiv.7.3-2.2336.

[9] K. Prokop, D. Połap, G. Srivastava, J. C.-W. Lin, Blockchain-based federated learning with checksums to increase security in internet of things solutions, Journal of Ambient Intelligence and Humanized Computing 14 (2023) 4685–4694. doi:10.1007/s12652-022-04372-0.

[10] M. M. Alshahrani, A secure and intelligent software-defined networking framework for future smart cities to prevent ddos attack, Applied Sciences 13 (2023) 9822. doi:10.3390/app13179822.

[11] Š. Grigaliūnas, M. Schmidt, R. Brūzgienė, P. Smyrli, S. Andreou, A. Lopata, Holistic information security management and compliance framework, Electronics. 13 (2024) 1–31. doi:10.3390/electronics13193955.

[12] M. Malik, M. Dutta, et al., Feature engineering and machine learning framework for ddos attack detection in the standardized internet of things, IEEE Internet of Things Journal 10 (2023) 8658–8669. doi:10.1109/JIOT.2023.3245153.

[13] J. Bhayo, S. A. Shah, S. Hameed, A. Ahmed, J. Nasir, D. Draheim, Towards a machine learning-based framework for ddos attack detection in software-defined iot (sd-iot) networks, Engineering Applications of Artificial Intelligence 123 (2023) 106432. doi:10.1016/j.engappai.2023.106432.

[14] Y. Su, D. Xiong, K. Qian, Y. Wang, A comprehensive survey of distributed denial of service detection and mitigation technologies in software-defined network, Electronics (2024). doi:10.3390/electronics13040807.

[15] E. Amadi, A. Ezeji, D. Okorie, I. Nnannandu, Volumetric, protocol and application layer ddos attack types on web servers, International Journal of Research (2016). DOI not available; article accessible at https://journals.pen2print.org/index.php/ijr/article/view/5627.

[16] Priyanka, D. Kumar, Decision tree classifier: a detailed survey, International Journal of Information and Decision Sciences (2020). doi:10.1504/IJIDS.2020.108141.

[17] T. Daniya, M. Geetha, K. S. Kumar, Classification and regression trees with gini index, Advances in Mathematics: Scientific Journal (2020). doi:10.37418/amsj.9.10.53.

[18] N. Sharma, S. Iqbal, Applying decision tree algorithm classification and regression tree (cart) algorithm to gini techniques binary splits, International Journal of Engineering and Advanced Technology (2023). doi:10.35940/ijeat.E4195.0612523.

[19] G. Biau, E. Scornet, A random forest guided tour, arXiv preprint arXiv:1511.05741 (2015). doi:10.48550/arXiv.1511.05741.

[20] L. Hu, J. Chen, J. Vaughan, H. Yang, K. Wang, A. Sudjianto, V. N. Nair, Supervised machine learning techniques: An overview with applications to banking, arXiv preprint arXiv:2008.04059 (2020). doi:10.48550/arXiv.2008.04059.

[21] Z. Li, H. Wang, Y. Zhang, X. Zhao, Random forest–based feature selection and detection method for drunk driving recognition, International Journal of Distributed Sensor Networks (2020). doi:10.1177/1550147720905234.