Egyptian Informatics Journal 24 (2023) 27–42

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full length article

Evaluation of Boruta algorithm in DDoS detection

Noor Farhana [a], Ahmad Firdaus [a,*], Mohd Faaizie Darmawan [b,*], Mohd Faizal Ab Razak [a]

[a] Faculty of Computing, College of Computing and Applied Sciences, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia
[b] Faculty of Computer & Mathematical Sciences, Universiti Teknologi Mara, Tapah, Perak, Malaysia

ARTICLE INFO

Article history:
Received 7 June 2022
Revised 4 October 2022
Accepted 14 October 2022
Available online 10 November 2022

Keywords:
DDoS
Boruta algorithm
Machine learning
J48
Random forest
Naïve bayes
Multilayer perceptron

ABSTRACT

Distributed Denial of Service (DDoS) is a type of attack that leverages many compromised systems or computers, as well as multiple Internet connections, to flood targeted resources simultaneously. A DDoS attack's main purpose is to disrupt website traffic and cause it to crash. As traffic grows over time, detecting a Distributed Denial of Service (DDoS) assault is a challenging task. Furthermore, a dataset containing a large number of features may degrade machine learning's detection performance. Therefore, in machine learning, it is necessary to prepare a relevant list of features for the training phase in order to obtain good accuracy performance. With far too many possibilities, choosing the relevant feature is complicated. This study proposes the Boruta algorithm as a suitable approach to achieve accuracy in identifying the relevant features. To evaluate the Boruta algorithm, multiple classifiers (J48, random forest, naïve bayes, and multilayer perceptron) were used so as to determine the effectiveness of the features selected by the the Boruta algorithm. The outcomes obtained showed that the random forest classifier had a higher value, with a 100% true positive rate, and 99.993% in the performance measure of accuracy, when compared to other classifiers.

1. Introduction

The DDoS is an attack which uses multiple distributed resources [1], for example, servers, services, or networks [2] against its targets. DDoS attacks are on the rise due to the increasing number of users or organizations using the Internet to exchange and deliver important data and information [3]. This tempts the attacker to make online services unavailable, or to stop legitimate users from accessing a specific network by overwhelming it with traffic from multiple sources [4]. The DDoS attack started in 1998, but people only became aware when it attacked corporations and organizations in July 1999 [5].One study noted that many organizations were attacked by the DDoS since the summer of 1999, and the numbers of attacks have been increasing. Due to this, research on DDoS attacks are performed continuously so as to find ways of preventing the attacks [6].Yahoo.com was one of the first organizations to be attacked by the DDoS in 2000, making the company's Internet services inaccessible, and a loss of advertising revenue for two hours [7]. In the third quarter of 2021, Kaspersky [8] reported that many waves of large-scale DDoS attacks many countries all over the world. According to the Incapsula study, the DDoS attacks come in various sizes and shapes, and they have not stopped growing [9]. About 86 % of the participants had reported of an average attack that lasts about 24 h or less. However, data showed that the lasting duration of the attacks was not consistent. Different organizations reported different time averages with 37 % of the organizations reporting an average of six hours or less, 31 % citing six to 12 h, and 18 % claiming 13 to 24 h. These events promote the expansion of DDoS threat.

Security practitioners conducted various experiments to detect DDoS. However, it is crucial to scrutinize the features that lead to the DDoS detection through a machine learning intelligent prediction model. The selection of features from the security event data or database will improve the performance of the machine learning detection [10,11,12]. The huge number of features in the dataset could decrease the performance of machine learning by slowing down the process of the training data, thereby making analysis less efficient [13,14,15,16].

* Corresponding authors.
E-mail addresses: mcn20001@stdmail.ump.edu.my (N. Farhana), firdausza@ump.edu.my (A. Firdaus), faaizie@uitm.edu.my (M.F. Darmawan), faizalrazak@ump.edu.my (M.F. Ab Razak).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2022.10.005

Various studies have been conducted on selecting DDoS features. A study [17], adopts Principal Component Analysis (PCA) approach selection by selecting a limited number of feature from the original sample. The following study uses Information gain and Chi-square techniques to select features to identifying DDoS attacks. Both [18] and [19] experiments adopts spark approach in cluster nodes for feature selection and classify DDoS threat. Despite the fact that many studies have been proposed various approaches to detect DDoS, a distinctive strategy is still required as DDoS attacks are rapidly expanding. DDoS attacks have not been completely handled by current DDoS analysis and detection methods [20]. As a result, as technology advances, DDoS detection methods need to be updated on a regular basis.

The main contribution of this study focuses on dynamic analysis that adopts minimal DDoS features to evaluate the effectiveness of Boruta algorithm. The research uses this algorithm to select relevant features among all available features in the dataset. In order to determine the feasibility of the Boruta algorithm, the study used four machine learning classifiers (J48, random forest, naïve bayes, and multilayer perceptron), and evaluate it after the feature selection. Furthermore, this article also evaluates performance of the classifiers in different DDoS dataset.

This study suggests using the Boruta algorithm to identify the pertinent features accurately. The performance of the features chosen by the Boruta algorithm and the Boruta algorithm's capacity to produce the best feature for machine learning detection were assessed using a variety of classifiers (J48, random forest, nave bayes, and multilayer perceptron). The results demonstrated that, in comparison to other classifiers, the random forest classifier had a higher value. The performance of machine learning tends to suffer from the high volume of features. From the tuning procedure and experiments carried out, it can be deduced that the Boruta algorithm offered better accuracy and could identify unidentified DDoS attacks by removing any unnecessary or unwanted data in the CICIDS2017 dataset, thereby permitting the machine learning to perform better as the Boruta algorithm is a wrapper built around the Random Forest classification. Its function is to capture all the interesting and important features in the dataset with the outcome variable. The Boruta will reject and eliminate the lower importance features. Then, in the classification phase, the Boruta features are used as an input during this process by splitting them into train and test datasets (70 % and 30 %) and continuing with the classification process using multiple classifiers. This phase is divided into two datasets of training and testing (70 % and 30 %) respectively to build the model and test the effectiveness of Boruta features in machine learning classification. Furthermore, this study aimed to measure the ability of the Boruta algorithm to achieve the best feature for machine learning detection and can be used for any advanced features for DDoS attack in the future. The analysis of the obtained results indicated that the model based on machine learning and the features selected by the Boruta algorithm gives a satisfactory feature selection quality with high accuracy, and thus allows us to get the best feature selection and predict less time consumption for this study compared with other studies.

The remainder of this paper is structured as follows. Section 2 surveys the related works. Section 3 provides the methodology. Section 4 presents the result, and Section 5 delivers limitation and future works, and followed by conclusion in Section 6.

2. Related work

This section starts by introducing the attacks of Denial of Service (DOS), explanation of Distributed Denial of Service (DDOS) attack detection methods, machine learning, Boruta algorithm and related previous research comparison on DDoS detection.

2.1. Attacks

There are many types of cyber-attacks on network infrastructures. These attacks can be based on confidentiality, integrity, and availability of the network packets, destinations, and sources [21]. The Denial of Service (DoS) attacks, and the Distributed Denial of Service (DDoS) attacks are some common attacks used by attackers because they are the most effective to flood a network or a server. Such attacks are performed so as to destroy, steal, or change the information or data in the systems. However, victims' confidential information cannot be stolen by this type of attack. Flooding the victims' computers with huge traffic is the main purpose of this attack since they cannot access the services provided by the server [22].

2.1.1. Denial of Service (DoS)

The Denial of Services (DoS) attacks use a single computer and a single Internet connection to flood a targeted resource or system. This attack will interrupt the network services, thereby leading to significant losses. The DoS attacks can be easily conducted and controlled by unskilled threat actors because of the uncomplicated steps. There are multiple types of DoS attacks, such as Volumetric attacks, SYN flooding, Fragmentation attacks, TCP-State exhaustion attack, Application Layer attacks, and Plashing [23].

2.1.2. Distributed Denial of Service (DDoS)

The Distributed Denial of Service (DDoS) attacks use multiple computer systems to interrupt the normal traffic of the targeted network, service, or server by flooding the systems with huge traffic. There are four components in the DDoS architecture, namely; Zombies or Bots, Botnet, Handlers, and Botmaster [24,25]. Initially, the Zombie was one of the components in the DDoS architecture which was a machine or computer that had been infected by malware. The actual attack is carried out by the Zombies by increasing traffic to the victims' computers of other machines significantly [6]. Bots are used as a design to infect a host. A group of bots that had been taken over, and controlled by the attacker is called a botnet. The handlers are the master commands who control the servers which control the group of Zombies. The botmaster will then control and handle all the botnets which flood the system.

2.2. DDoS attack detection method

2.2.1. Signature-Based method

The signature-based methodology is a human-dependent process. It requires several hours of testing, creating, and deploying of the signature. It monitors and compares the connections or network packets with predetermined patterns [26], and is effective against known attacks [27].It is also simple and efficient at processing audit data. However, this method cannot detect novel anomalies that are not defined in the signatures [26]. Instead, the signature system needs to be frequently updated by the administrator.

2.2.2. Anomaly-Based method

The anomaly-based methodology will create a baseline profile of the normal network, program, or system [26]. It also will help in implementing a system that can learn from data. The unseen data also will provide a prediction based on the data that had been learned. There are several advantages regarding this method such as the ability to detect unknown and new attacks (zero-day) [27].All events detection of malfunctioning of the protected web-server whether the events are malicious or not is one of the advan-

tages of this method. Besides, without setting off an alarm there will be a problem for an attacker to detect the certain activity that can be carried out [26]. However, this method also requires training for a specific web server is one of the weaknesses of this method and a trained classifier possibly will not work properly with other or same web servers that have different hardware and software components. The categories of anomaly-based are soft computing, knowledge-based / cognitive-based, and machine learning.

2.2.3. Soft computing

This method identifies the DDoS attacks by using various types of soft computing, such as Fuzzy reasoning [27], Artificial Neural networks, and others. This method is mostly performed [28] for known and supervised attacks. It uses a collection of optimization and processing styles that allow inexactness and uncertainty to be identified [29,30].

2.2.4. Knowledge-based / cognitive based

This method is able to analyze, detect, and extract network events of the pattern or system vulnerabilities shown by the attack. The knowledge-based method is broadly classified as expert system, state transition analysis, signature analysis, and self-organizing maps. This method can be used to source victims' network [30,31]. The only issue arising from this method is that it needs a detailed analysis during the task [31].

2.2.5. Machine learning-based method

Learning and improving the performance of the group or specific task is characterized as a framework, or capacity program of the machine learning-based method [31]. This method can detect features, such as packet size, packet rate, bit rate, and others. The previous results generated from this method can be used as a strategy to build a framework. The execution of this strategy can be changed based on newly acquired data. The machine learning-based method can be broadly classified as genetic algorithms, Bayesian approaches, neural networks, support vector machines, and fuzzy logic [32,33]. One of its advantages is its ability to capture interdependencies, adaptability, and flexibility [31,34].

2.3. Boruta algorithm

The Boruta algorithm was invented by Witold Rudnicki, and Miron Kursa, Polish researchers from the University of Warsaw. This algorithm works as a wrapper around Random Forest. Its function is to capture all the interesting and important features in the dataset with the outcome variable.

In contrast, most traditional feature elimination method follows a minimal optimal method that often relies on a small subset of features which yield the smallest error on a chosen classifier, and also mislay some features. To measure the important attributes or features, the average drop accuracy of Zscore is used, and the fluctuations of the average precision loss of trees in the forest is counted.

A mixed shadow feature set is created to eliminate the correlation between the predicted value and the features before the selection starts. The Boruta algorithm consists of the following [35,36,37]:

- Initially, the data set is extended to create duplicate copies of all the independent variables so as to establish the hybrid features ($N = [M, P]$). This is done by randomly scrambling the matrix of the real sample feature which is M so as to create P as a shadow feature.
- Permuted copies or shadow features are the disorder mixed features set by random shuffling which eliminates the correlation between response variables and features.
- Shuffled copies and the originals are then combined. A random forest classifier is then established to calculate the average reduced accuracy, Z value, and the importance of all the features, with the more important features being calculated based on the highest value of z. The Zmax is recorded as the highest Zvalue in the shadow features.
- The result of the important judgement features is then calculated based on the result of the Zvalue, where if zvalue > Zmax, the features will be considered as "Important" and "Retained" while if zvalue < Zmax, the features will be considered as an "Unimportant" and "Deleted."
- This process is repeated until all the maximum number of iterations are reached by the algorithm or stopped until all the features are rejected or confirmed.

2.4. Machine learning

2.4.1. Unsupervised learning

Unsupervised learning occurs by using unlabelled data to discover and analyze the pattern and trends for association and clustering problems. However, this learning also allows the model to learn more about the data, and to understand many structure or distribution in the data. This learning will perform more complex tasks since it depends on the model to work on its own.

2.4.2. Supervised learning

Supervised learning learns and uses labelled datasets to train data for the prediction and classification of the outcomes accurately, for unforeseen data. However, this learning will teach models to give the perfect outcome. Two types of supervised learning are classification and regression.

2.4.3. Classification

This algorithm will fit accurately test dataset into specific types and categories as a training dataset and will recognize every single entity specifically for relevant and accurate outcomes. Hence, support vector machines (SVM), decision tree, random forest, linear classifiers, and k-nearest neighbors are the common classification algorithms.

2.4.4. Regression

This algorithm is used to understand and recognize the relationship between the independent and dependent variables. The various types of regression algorithms include polynomial regression, logistical regression, and linear regression.

In the following, we briefly describe the algorithms used in our experiments.

2.4.5. Random forest

Random Forest is the most famous classification model in machine learning. This classifier has a huge number of decision trees (DT) which consist of two types of nodes - child and parent nodes. This classifier was developed by Adele Cutler and Leo Breiman who combined the decision trees for predicting new unlabeled data [34,38]. Scikit-learn was implemented for each decision tree by calculating the importance of a node using the Gini Importance. Here two child nodes (binary tree) are assumed:

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

Equation 1 Calculation a nodes importance using Gini Importance in Scikit-Learn.

- ni sub(j) = the importance of node j
- w sub(j) = weighted number of samples reaching node j
- C sub(j) = the impurity value of node j
- left(j) = child node from left split on node j
- right(j) = child node from right split on node j
- sub() is being used as subscript isn't available in Medium

$$fi_i = \frac{\sum j : nodejsplitsonfeatureini_j}{\sum k \in allnodesni_k}$$

Equation 2 Calculation of a the importance of feature.

- fi sub(i) = the importance of feature i
- ni sub(j) = the importance of node j

These can then be normalized to a value between 0 and 1 by dividing it with the sum of all feature importance values:

$$normfi_i = \frac{fi_i}{\sum j \in allfeaturesfi_j}$$

Equation 3 Calculation of a normalized the importance of feature.

The final feature importance, at the Random Forest level, is its average over all the trees. The sum of the feature's importance value on each tree is calculated, and divided by the total number of trees:

$$RFfi_i = \frac{\sum j \in allfeaturesfi_j}{T}$$

Equation 4 Calculation of the importance of feature i calculated from all trees in the Random Forest model.

- RFI sub(i) = the importance of feature i calculated from all trees in the Random Forest model
- normfi sub(ij) = the normalized feature importance for i in tree j
- T = total number of trees

2.4.6. j48

A decision tree is generated in the WEKA, for example, ID3 (Iterative Dichotomiser 3) which was built by using a set of training data [39,40]. The C4.5 algorithm is used to build a decision tree by implementing J48 as a classifier. Besides being accurate in prediction, this classifier can also help to explain the patterns. It can easily deal with missing values, estimate error rate, pruning, and generating rules from the tree. However, it has numeric attributes, and complexity in the induction of the decision tree [41]. This classifier can handle discrete and continuous attributes [42]. The internal nodes of a decision tree denote the different attributes while the branches between the nodes denote the possible values that these attributes can have in the observed samples. The terminal nodes denote the final values (classification) of the dependent variable [43].

INPUT.

- Training Dataset = D.

OUTPUT.

- Decision Tree = T.

DTBUILD (*D).

- T = φ;
- T = Create root node and label with splitting attribute.
- T = Add arc to root node for each split predicate and label;
- For each arc D = Database created by applying a splitting predicate to D;
  - If stopping point reached this path, then T'= create a leaf node
  - and labeled with appropriate class;
  - or else T'= DTBUILD(D); T = add T' to arc;

2.4.7. Multilayer perceptron (MLP)

The Neural network is a set of algorithm that is modelled after the human brain to recognize patterns. The MLP is an implementation for neural network [41]. It can be used to solve a difficult, and complex task in machine learning. There are three layers to run data through, such as input layer, hidden layer, and output layer. The backpropagation or supervised learning technique is utilized by Multilayer Perceptron to train the network [44].

The MLP is an algorithm that can distinguish data that are not linearly separable. It is made up of more than one perceptron, with an input layer that receives the signal. The hidden layers are the true computational engine of the MLP while the output layer makes a prediction or decision about the input. The MLP utilizes a supervised learning technique called backpropagation for training. It relates the weight and bias to the error so that it can be measured in many ways. For the weight of the neuron to be updated, it uses this equation[45]:

$$\text{weight} = \text{weight} + \text{learning\_rate} * (\text{expected} - \text{predicted}) * x$$

Equation 5 calculation of the neuron weight.

The feed forward neural network is supplemented by the multilayer perceptron (MLP). The input layer, output layer, and hidden layer are the three different types of layers that contribute to making up the entire system.

Fig. 1 shows the interconnected layers of the multilayer perceptron the input layer, the hidden layers which may have more than one layer, and the output layer. The input layer contains input neurons that send information to the hidden layer. The data from the hidden layer will then be sent to the output layer. Every neuron has weighted inputs called synapses, which are the parameters that will convert a neural network to a parameterized system, an activation function, and one output. To obtain one output from the neuron, the activation signal produced by the weighted sum of the inputs will be passed to the activation function. One advantage

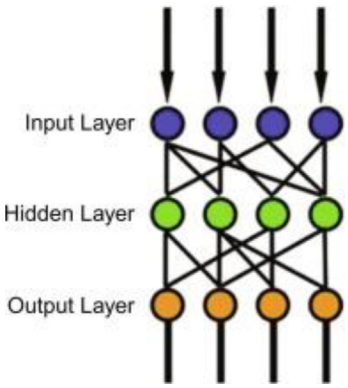

Fig. 1. Scheme of a three-layer MLP.

of the MLP is that it minimizes the prediction error of one or more target variables. The MLP algorithm is good for mapping and regression. It is also capable of generalization where an unknown pattern is classified into known patterns that share the same features [44]. Even if a significant fraction of its neurons and interconnection fails, it can still work, and relearn after the damage. Its disadvantage is that during the training process, the stop time cannot be guaranteed. At this time, if the user sets the number of hidden neurons to low value, the model may become underfitting while if set at high value, the model may be overfitting.

2.4.8. Naïve bayes

Another classification method that can be used is the naïve bayesian method also called a multi-label problem. This method can be used for continuous and discrete attributes. This classification method is used for nonlinear and multi-label problems which learn the conditional class of the model [38,34]. It is effective for big data in the dataset. The bayes theorem finds the probability of an occurring event given the probability of another event that has already occurred. Below is the naïve bayes equation to separate attack traffic from the normal traffic [46].

$$f_{i(X)=\prod_{j=1}^{N} P(X_j|C_i)P(c_i)}$$

Equation 6 Equation for naïve bayes.

where X = (x_1, x_2, . . ., x_N) denotes a feature vector and j = 1, 2, . . ., N, denote possible class labels. The training phase for learning a classifier consists of estimating conditional probabilities P (xj\ ci), and prior probabilities P (ci). Here, P(ci) is estimated by counting the training examples that fall into class ci, and then dividing the resulting count by the size of the training set.

To build a classifier, one needs to find the probability of a given set of input for all possible values of the class variable y, and then to pick the output with maximum probability. This can be expressed mathematically as:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(X_i|y)$$

Equation 7 Equation classification rules.

We have explained multiple machine learning algorithms (J48, random forest, naïve bayes, and multilayer perceptron). However, in machine learning, it is critical to select the features in order to construct a stable machine learning model. Excessive features will increase the dimensionality of datasets, irrelevant data, time, expense of tests, and, most significantly, lower detection accuracy performance [11,47]. Therefore, in this study, we adopt Boruta algorithm for feature selection algorithm.

2.5. Previous research comparison on DDoS detection

Table 1 compares this study to previous DDoS studies that adopt different approaches for feature selection. The comparison tabulates that the current study is the only one that examine into the Boruta algorithm in selecting features and investigate its performance in machine learning detection. The following section describes the methodology on this approach in detail.

3. Methodology

To investigate the effectiveness of Boruta algorithm in selecting features, we applied multiple machine learning classifiers (J48, random forest, naïve bayes, and multilayer perceptron) to classify between DDoS and normal activities. The steps are as follow:

Table 1
Previous DDoS studies comparison on feature selection.

References	Year	Feature selection approach
[18]	2022	Spark cluster
[19]	2020	Spark streaming
[6]	2011	Information gain and Chi-square
[17]	2010	Principal Component Analysis (PCA)
This study		Boruta algorithm

3.1. Phase 1: Requirement planning

Fig. 2 illustrates the workflow of our methodology. This phase reviewed the DDoS attack detection methods and algorithms in previous research. Followed by the comparison of the advantages and disadvantages of each method, so as to discover the research gap which can be applied to identify the new output from the implementation. This is imperative to have a better accuracy of detecting the DDoS attacks. The Boruta algorithm is suggested in this study as a suitable method to achieve accuracy in identifying the pertinent features. In order to evaluate the effectiveness of the features selected from the Boruta algorithm, several classifiers (J48, random forest, naïve bayes, and multilayer perceptron) were used.

3.2. Phase 2: Research implementation and method development

Fig. 3 exhibits the research implementation that begins with data collection, data cleaning, feature selection and classification.

3.2.1. Data collection

In phase 2, [48]. This is a new IDS dataset for network security, and intrusion detection purposes. The dataset was provided by the University of New Brunswick, and it contained 79 features and 225,745 instances, and two class labels including benign and DDoS attack traffic. The dataset also contained the network traffic during working hours. It comes in eight CSV files which separate all the features and instances into rows and columns. Data capturing started at 9 a.m. on 3 July 2017 (Monday), and it ended at 5p.m. on 7 July 2017 (Friday), a total of five days. During this period, several types of attacks were launched, for instances, the DDoS, Infiltration DoS, Heartbleed, Web Attack, Brute Force SSH, Infiltration, Botnet attack, and Brute Force FTP. This study, however, only uses the Friday afternoon (03:56p.m. until 04:16p.m.) dataset which contained the DDoS LOIT traffic for depth analysis traffic. Table 2 highlights the samples of the dataset which used the Low Orbit Ion Cannon (LOIC) as a DDoS attack tool. This was achieved by sending the HTTP, UDP, and TCP requests to the target server [49]. Once we have identified the exact dataset, the following step, is the data cleaning process.

3.2.2. Data cleaning

This process is needed to check and eliminate all the NAS input, and zero values in the dataset. All the raw data were transformed into persistent data for ease of analysis so as to prevent poor performance in machine learning. The processing time will expedite if there was no disruption in the dataset. Hence, all the unnecessary data would be removed from the dataset using RStudio. Fig. 4 lists the total of four rows (rows 6797, 14740, 15048, and 209729) which contained unnecessary data, thus the initial rows, 225,745 became 225,741 in the dataset (Fig. 4). Once we obtained the clean dataset, the next phase implements the Boruta algorithm for feature selection (Fig. 5).

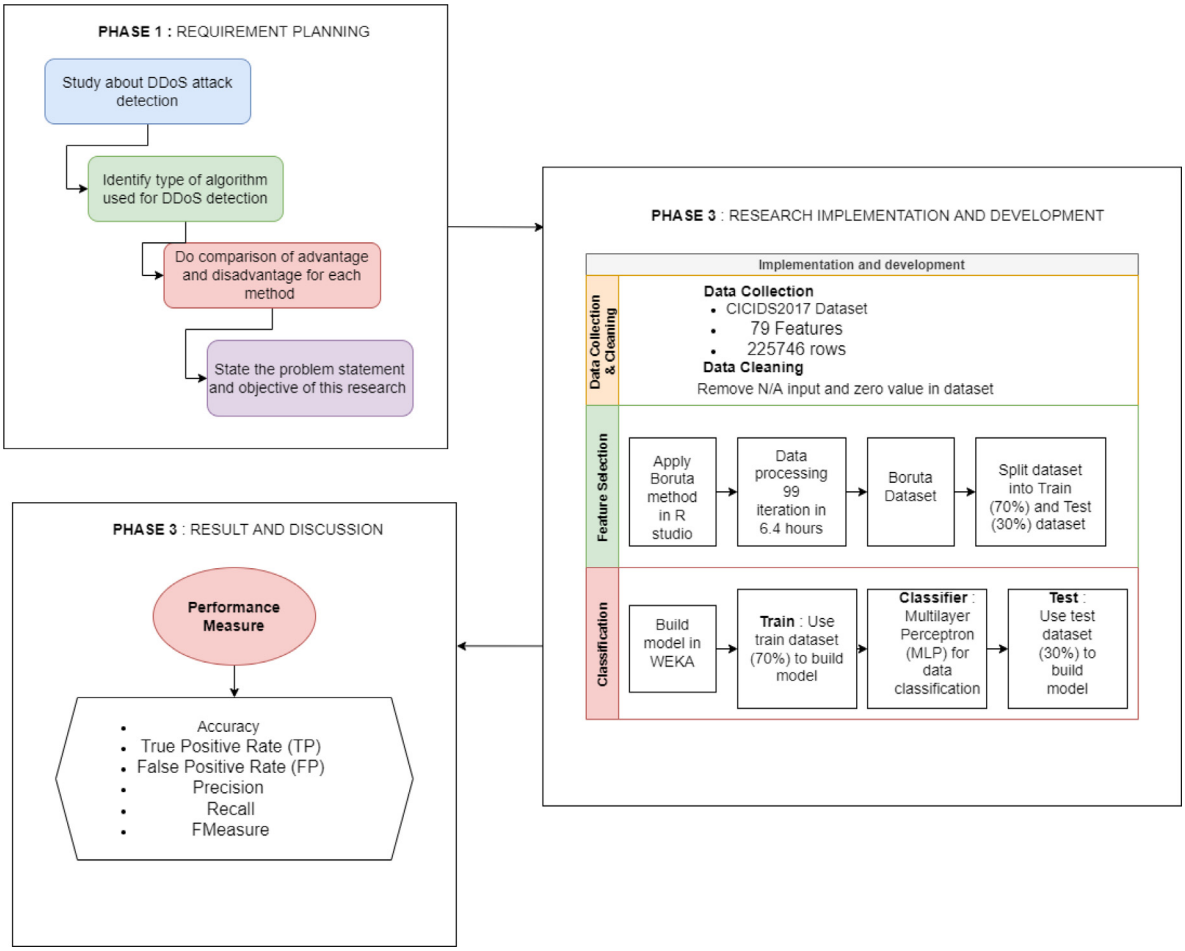

Fig. 2. Architecture of workflow.

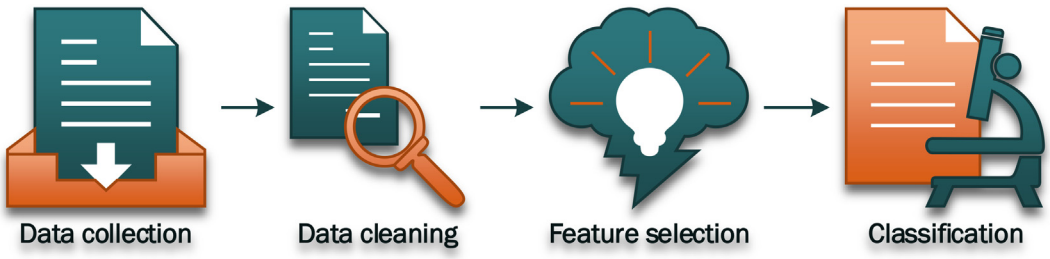

Fig. 3. Research implementation.

Table 2
Traffic explanation of CICIDS2017 dataset.

Day	Labels
Monday	Benign (normal)
Tuesday	Brute Force, SFTP, and SSH
Wednesday	DoS and Heartbleed attack Slowloris, Slowhttptest, Hulk, and GoldenEye
Thursday	Brute Force, XSS, and SQL Injection
Friday	The DDoS attack, Botnet ARES, Portscan

3.2.3. Features selection

This phase is a process of manually or automatically selecting those features which contribute to the prediction variable or output. The presence of any irrelevant data could decrease the accuracy of the model. Three types of feature selection were made: filter, wrapper, and embedded.

This study used Boruta algorithm for selecting the features. This algorithm is a wrapper built around the Random Forest classifica-

```
> which(!complete.cases(CICIDS2017))
[1]   6797  14740  15048 209729
```

Fig. 4. List of rows need to eliminate.

```
225745 obs. of 79 variables
225741 obs. of 79 variables
```

Fig. 5. Number of rows after the elimination process.

```
 11. run of importance source...
Growing trees.. Progress: 30%. Estimated remaining time: 1 minute, 13 seconds.
Growing trees.. Progress: 60%. Estimated remaining time: 40 seconds.
Growing trees.. Progress: 93%. Estimated remaining time: 6 seconds.
Computing permutation importance.. Progress: 20%. Estimated remaining time: 2 minutes, 4 seconds.
Computing permutation importance.. Progress: 41%. Estimated remaining time: 1 minute, 29 seconds.
Computing permutation importance.. Progress: 61%. Estimated remaining time: 58 seconds.
Computing permutation importance.. Progress: 83%. Estimated remaining time: 25 seconds.
 12. run of importance source...
Growing trees.. Progress: 33%. Estimated remaining time: 1 minute, 4 seconds.
Growing trees.. Progress: 65%. Estimated remaining time: 32 seconds.
Computing permutation importance.. Progress: 20%. Estimated remaining time: 2 minutes, 0 seconds.
Computing permutation importance.. Progress: 42%. Estimated remaining time: 1 minute, 25 seconds.
Computing permutation importance.. Progress: 64%. Estimated remaining time: 52 seconds.
Computing permutation importance.. Progress: 86%. Estimated remaining time: 19 seconds.
 13. run of importance source...
Growing trees.. Progress: 37%. Estimated remaining time: 53 seconds.
Growing trees.. Progress: 74%. Estimated remaining time: 21 seconds.
Computing permutation importance.. Progress: 21%. Estimated remaining time: 1 minute, 58 seconds.
Computing permutation importance.. Progress: 42%. Estimated remaining time: 1 minute, 24 seconds.
Computing permutation importance.. Progress: 64%. Estimated remaining time: 51 seconds.
Computing permutation importance.. Progress: 85%. Estimated remaining time: 21 seconds.
After 13 iterations, +55 mins:
 confirmed 60 attributes: ACK.Flag.Count, act_data_pkt_fwd, Active.Std, Average.Packet.Size, Avg.Bwd.Segment.Size and 55 more;
 rejected 12 attributes: Bwd.Avg.Bulk.Rate, Bwd.Avg.Bytes.Bulk, Bwd.Avg.Packets.Bulk, Bwd.PSH.Flags, Bwd.URG.Flags and 7 more;
 still have 6 attributes left.
```

Fig. 6. Result for first 13 iterations.

```
 29. run of importance source...
Growing trees.. Progress: 33%. Estimated remaining time: 1 minute, 3 seconds.
Growing trees.. Progress: 67%. Estimated remaining time: 30 seconds.
Computing permutation importance.. Progress: 24%. Estimated remaining time: 1 minute, 37 seconds.
Computing permutation importance.. Progress: 49%. Estimated remaining time: 1 minute, 4 seconds.
Computing permutation importance.. Progress: 75%. Estimated remaining time: 30 seconds.
 30. run of importance source...
Growing trees.. Progress: 32%. Estimated remaining time: 1 minute, 4 seconds.
Growing trees.. Progress: 67%. Estimated remaining time: 31 seconds.
Computing permutation importance.. Progress: 25%. Estimated remaining time: 1 minute, 34 seconds.
Computing permutation importance.. Progress: 50%. Estimated remaining time: 1 minute, 1 seconds.
Computing permutation importance.. Progress: 76%. Estimated remaining time: 29 seconds.
 31. run of importance source...
Growing trees.. Progress: 32%. Estimated remaining time: 1 minute, 5 seconds.
Growing trees.. Progress: 66%. Estimated remaining time: 31 seconds.
Computing permutation importance.. Progress: 25%. Estimated remaining time: 1 minute, 34 seconds.
Computing permutation importance.. Progress: 50%. Estimated remaining time: 1 minute, 1 seconds.
Computing permutation importance.. Progress: 76%. Estimated remaining time: 30 seconds.
After 31 iterations, +2 hours:
 confirmed 2 attributes: Bwd.IAT.Min, Fwd.PSH.Flags;
 still have 4 attributes left.
```

Fig. 7. Result for first 31 iterations.

```
> print(final_boruta)
Boruta performed 99 iterations in 6.408809 hours.
Tentatives roughfixed over the last 99 iterations.
 64 attributes confirmed important: ACK.Flag.Count, act_data_pkt_fwd, Active.Min, Active.Std,
Average.Packet.Size and 59 more;
 14 attributes confirmed unimportant: Active.Max, Active.Mean, Bwd.Avg.Bulk.Rate, Bwd.Avg.Bytes.Bulk,
Bwd.Avg.Packets.Bulk and 9 more;
```

Fig. 8. Result for 99 iterations.

Table 3
Summary of Boruta method result.

Phase	Confirmed important	Confirmed unimportant	Attribute in progress	Iteration	Time taken
1st Phase	60 attributes	12 attributes	6 attributes	13	55 mins
2nd Phase	62 attributes	0 attributes	4 attributes	31	2 h
Final Phase	64 attributes	14 attributes	0 attributes	99	6.408809 h

tion. Its function is to capture all the interesting and important features in the dataset with the outcome variable.

This study proposes to use dual feature selections to minimise the relevant features as much as possible to enhance the effectiveness of machine learning when detecting the DDoS attacks in network traffic.

In the first layer of feature selection, the Boruta method acts as the main actor for selecting the best features in the dataset by using RStudio. The Boruta is a wrapper algorithm which can remove any unnecessary or unwanted data in the CICIDS2017 dataset, thereby permitting the machine learning to perform better. The Boruta will reject and eliminate the lower importance features.

Fig. 6 shows that 60 attributes were confirmed while 12 attributes were rejected in 55 min for the first 13 iterations. This left 12 attributes to be run in the iterations. After 31 iterations (Fig. 7), the result showed that two attributes were confirmed, and four attributes remained in two hours. In Fig. 8, after 99 itera-

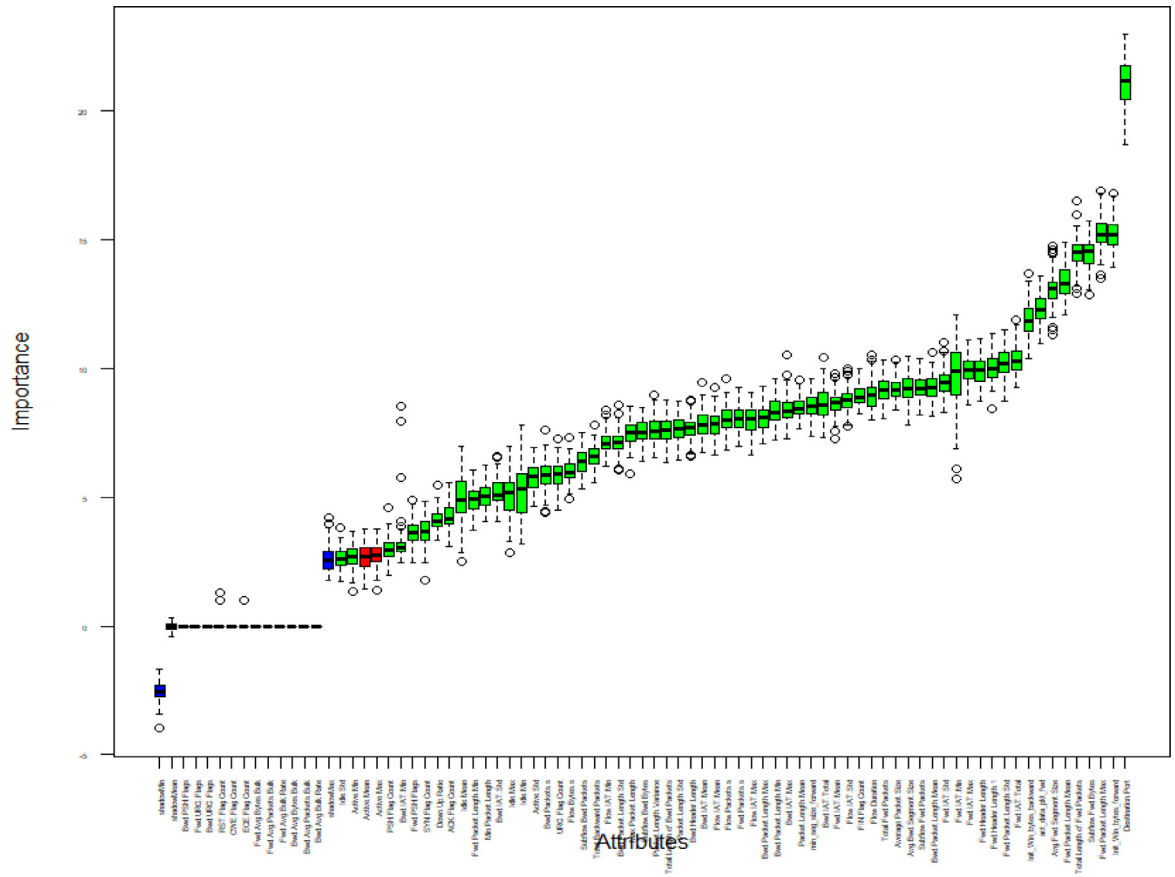

Fig. 9. Plot of Boruta method result.

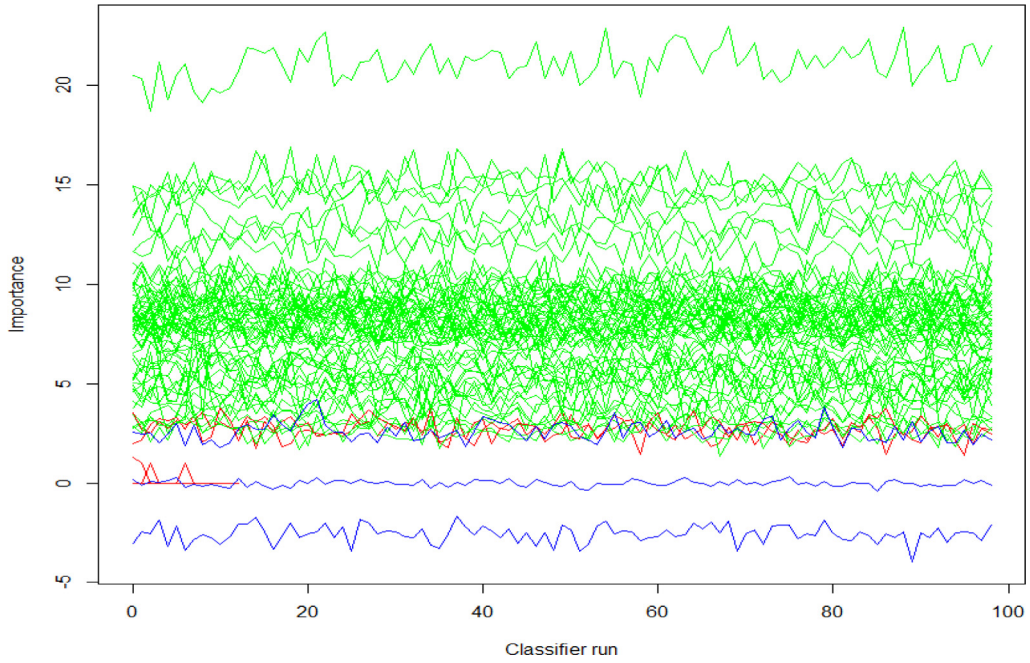

Fig. 10. Plot of Boruta method result.

tions, the result confirmed that 64 attributes were important, and 14 attributes were unimportant, following six hours for Boruta takes to calculate. Table 3 lists the summary of the Boruta progress in different iterations, whereas 99 iteration takes the longest hours (6 hours) than other iterations.

a. Plot analysis of relative importance ranking of *Z_score* for each feature.

As depicted in Fig. 9 and Fig. 10, *Z_score* was used to measure the importance of each feature. The blue box plot represents the

Table 4
Importance features or attributes.

Important Attributes			
Group	Features	Group	Feature
Active	• Active.Min Active.Std	**Fwd.Packet**	• Fwd.Packet.Length.Max Fwd.Packet.Length.Mean Fwd.Packet.Length.Min Fwd.Packet.Length.Std Fwd.Packet.s
Average	• Average.Packet.Size Avg.Bwd.Segment.Size Avg.Fwd.Segment.Size	**Flag.Count**	• PSH.Flag.Count SYN.Flag.Count URG.Flag.Count FIN.Flag.Count ACK.Flag.Count
Bwd.IAT	• Bwd.IAT.Max Bwd.IAT.Mean Bwd.IAT.Min Bwd.IAT.Std Bwd.IAT.Total	**Idle**	• Idle.Max Idle.Mean Idle.Min Idle.Std
Bwd.Packet	• Bwd.Packet.Length.Max Bwd.Packet.Length.Mean Bwd.Packet.Length.Min Bwd.Packet.Length.Std Bwd.Packet.s	**Init_Win_bytes**	• Init_Win_bytes_backward Init_Win_bytes_forward
Flow	• Flow.Bytes.s Flow.Duration Flow.Packets.s	**Packet.Length**	• Max.Packet.Length Min. Packet.Length Packet.Length.Mean Packet.Length.Std Packet.Length.Variance
Flow IAT	• Flow.IAT.Max Flow.IAT.Mean Flow.IAT.Min Flow.IAT.Std	**Subflow**	• Subflow.Bwd.Bytes Subflow.Bwd.Packets Subflow.Fwd.Bytes Subflow.Fwd.Packets
Fwd.Header	• Fwd.Header.Length Fwd.Header. Length1	**Total**	• Total.Backward.Packets Total.FWD.Packets Total.Length. of.BWD.Packets Total.Length. of.FWD.Packets
Fwd.IAT	• Fwd.IAT.Max Fwd.IAT.Mean Fwd.IAT.Min Fwd.IAT.Std Fwd.IAT.Total	**Other**	• Min_seg_size_forward Destination.Port Down.Up.Ratio Fwd.PSH.Flags Act_data_pkt_fwd Bwd.Header.Length
Total		**64 Features**	

Table 5
Unimportant features or attributes.

Unimportant Attributes	
• Fwd.Avg.Bytes.Bulk	• Active.Max
• Fwd.Avg.Packets.Bulk	• Fwd.URG.Flags
• Fwd.Avg.Bulk.Rate	• Bwd.URG.Flags
• Bwd.Avg.Bytes.Bulk	• RST.Flag.Count
• Bwd.Avg.Packets.Bulk	• CWE.Flag.Count
• Bwd.Avg.Bulk.Rate	• ECE.Flag.Count
• Bwd.PSH.Flag	• Active.Mean
Total	**14 Features**

Table 6
Evaluation information.

Performance measure	**Definition**	**Equation**	**Description**
Accuracy	Achievement of the correct result.	$\frac{TP+TN}{(TP+TN+FP+FN)}$	
True Positive (TP) Rate	True Positive rate Instances correctly classified as a given class	TP	
Precision (P)	the proportion of instances that are true of a class	$\frac{TP}{(TP+FP)}$	Greater value shows, excellent performance
Recall (R)	equivalent to TP rate	$\frac{TP}{(TP+FN)}$	
F-measure	A combined measure for precision and recall calculated as 2 * Precision * Recall / (Precision + Recall)	$\frac{2PR}{(P+R)}$	
False Positive (FP) Rate	False Positive Rate instances falsely classified as a given class	FP	Lower value shows, excellent performance

shadow attributes corresponding to the maximum, average, and minimum of *Z_score* results. The green and red box plots represent the *Z_score* for the confirmation, and refusal attributes. The plot analysis below shows that the variables in the red color of a box plot and line graph represent the unimportant attributes. Meanwhile, the variables in the green color of the box plot and line graph represent the important attributes.

b. List of relative importance for each feature.

Table 4 highlights the list of important attributes detected after the Boruta method process was performed in RStudio. A total of 64 features were selected and classified as important attributes after 99 iterations which consumed more than six hours.

Table 5 presents the list of unimportant attributes after the Boruta method process was run in RStudio. A total of 14 features were classified as unimportant attributes after 99 iterations which took more than six hours to run.Table 6.

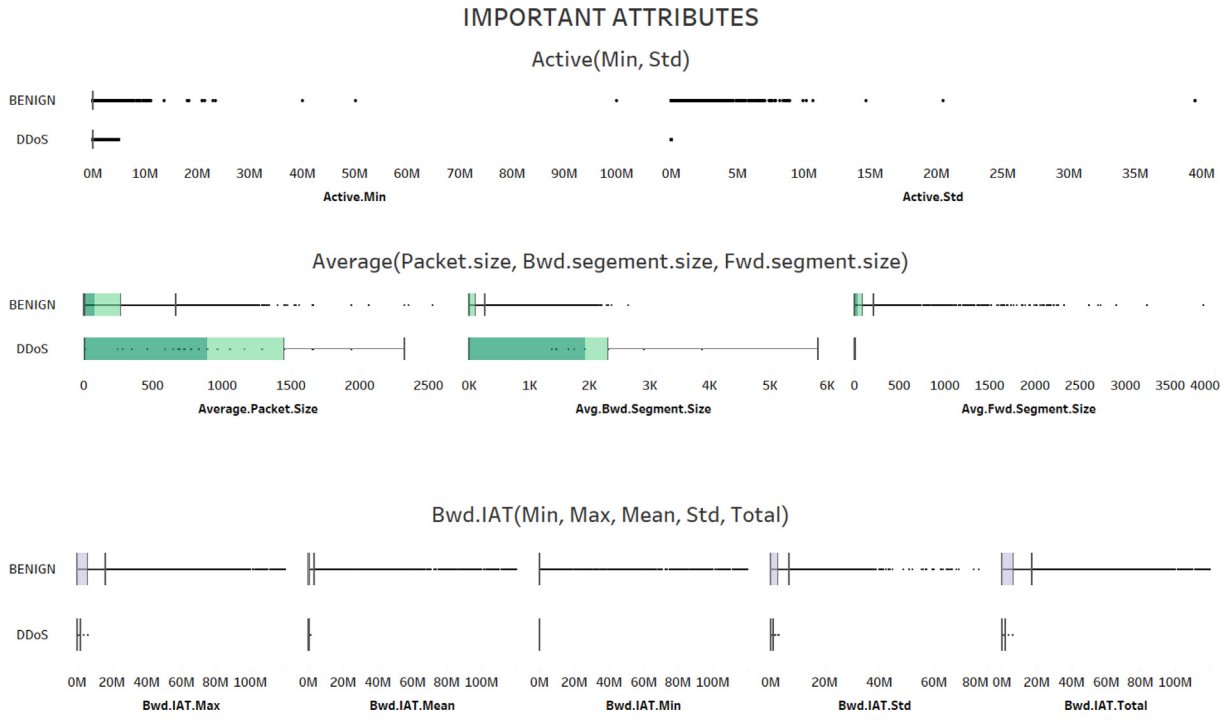

Fig. 11. Important attributes for (Active, Average and Bwd.IAT) in visualization.

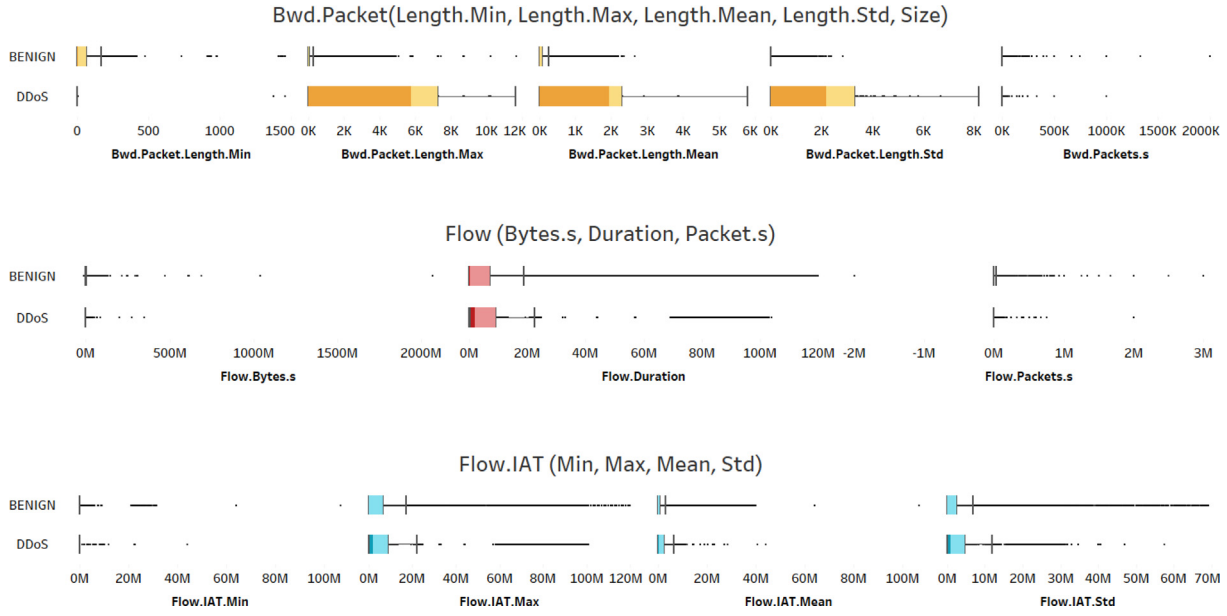

Fig. 12. Important attributes for (Bwd.Packet, Flow and Flow.IAT) in visualization.

c. Important features or attributes in visualization

The following figures from Fig. 11 until Fig. 16, illustrate the important features using visual analytic in Tableau. A total of 64 features were presented in graphical views. Visualization or visual analytic is excellent in searching hidden behaviors or patterns for certain features by using various types of graphical views. It operates by capturing all the interesting and important features in the dataset for a clear and better view. Following the Boruta algorithm, the box plot analysis was then applied to visualize all the results of the important features or attributes. All the box plots in all figures (Fig. 11 until Fig. 16) show significant difference and clear gap between benign (normal activities) and DDoS attacks. For instance, the color and outliers of the box plots in DDoS extend more than benign (Figs.12-15).

3.2.4. Classification

Classification is a supervised learning approach where the computer program will learn from the data input that has been given to it and then will use this learning to classify new observations [50]. In this phase, the Boruta features is used as an input during this process by splitting into train and test datasets (70 % and 30 %) and continue with the classification process using multiple classifiers (J48, random forest, naïve bayes, and multilayer

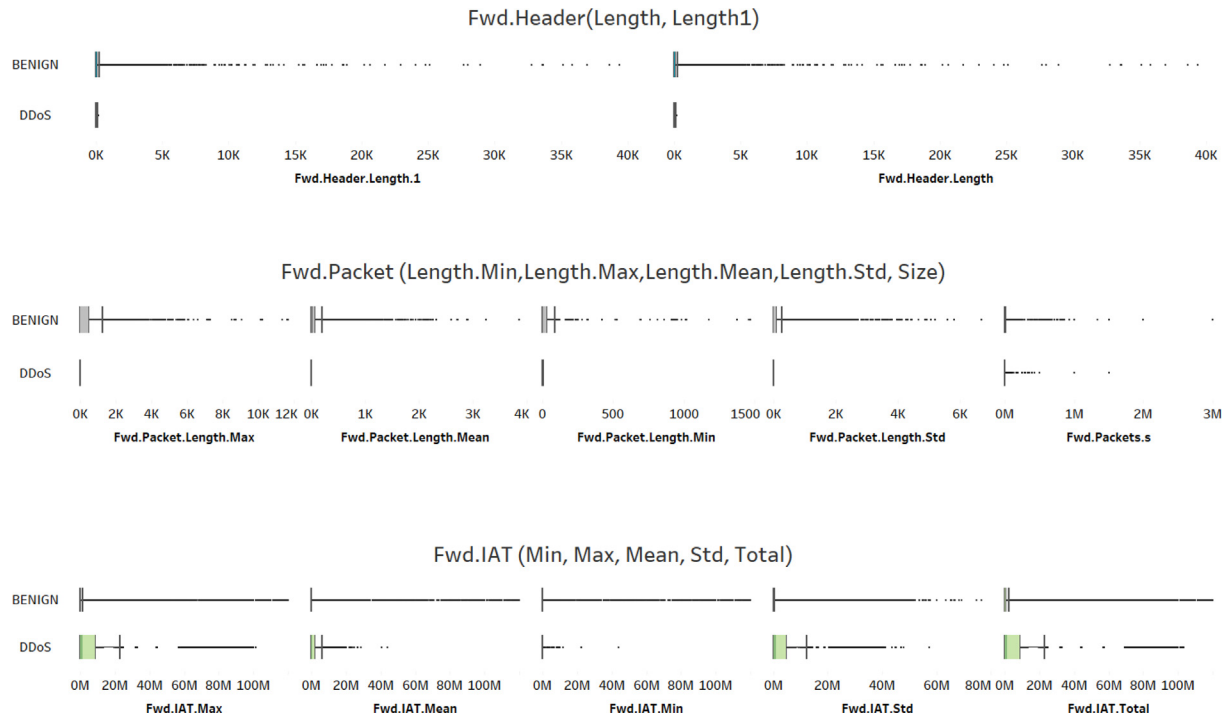

Fig. 13. Important attributes for (Fwd.Header, Fwd.Packet and Fwd.IAT) in visualization.

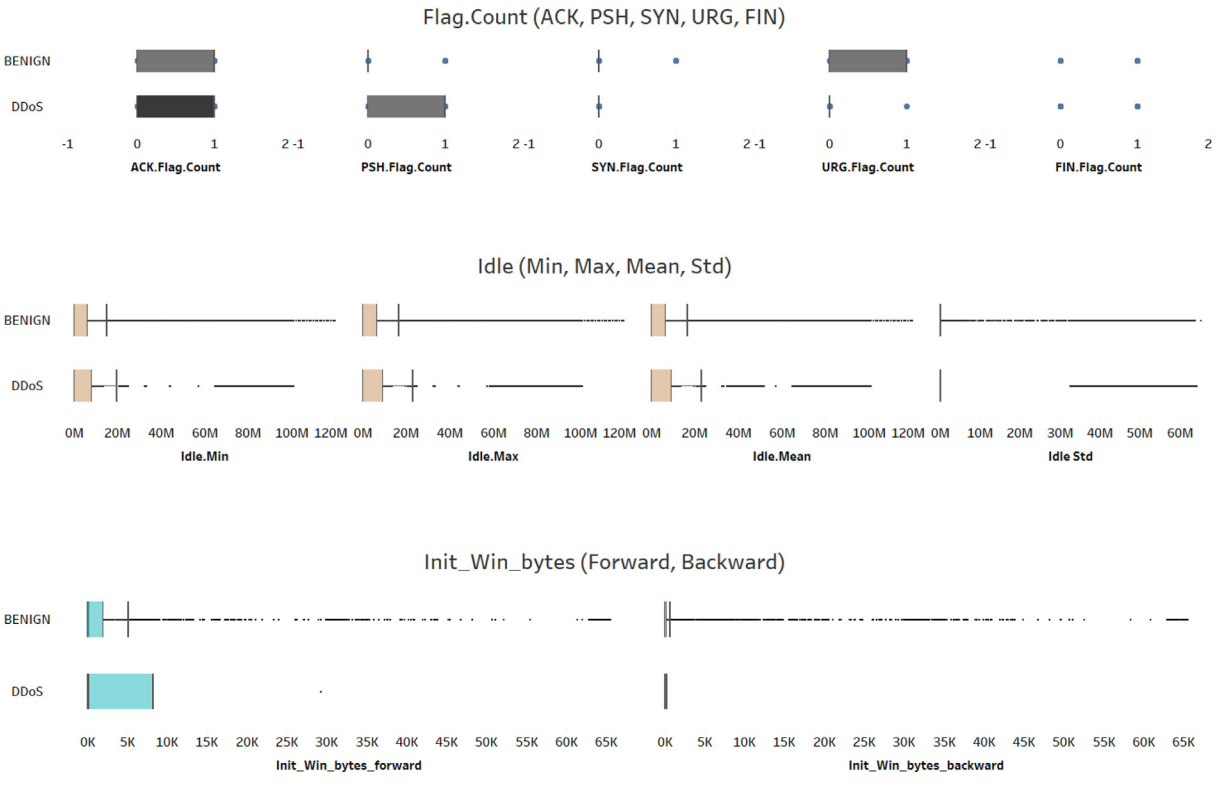

Fig. 14. Important attributes for (Flag.Count, Idle and Ini_Win_bytes) in visualization.

perceptron). This phase divided into two datasets of training and testing (70 % and 30 %) respectively to build the model and test the effectiveness of Boruta features in machine learning classification.

4. Results

This section discussed the results that consist of multiple performance evaluations. The terms of performance evaluation types

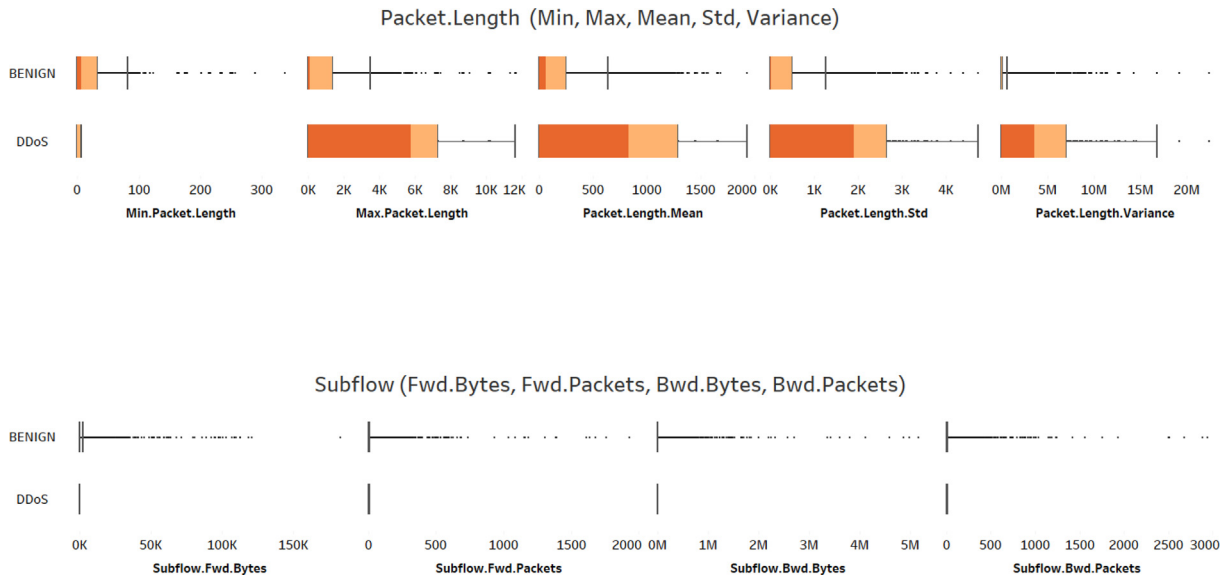

Fig. 15. Important attributes for (Packet.Length and Subflow) in visualization.

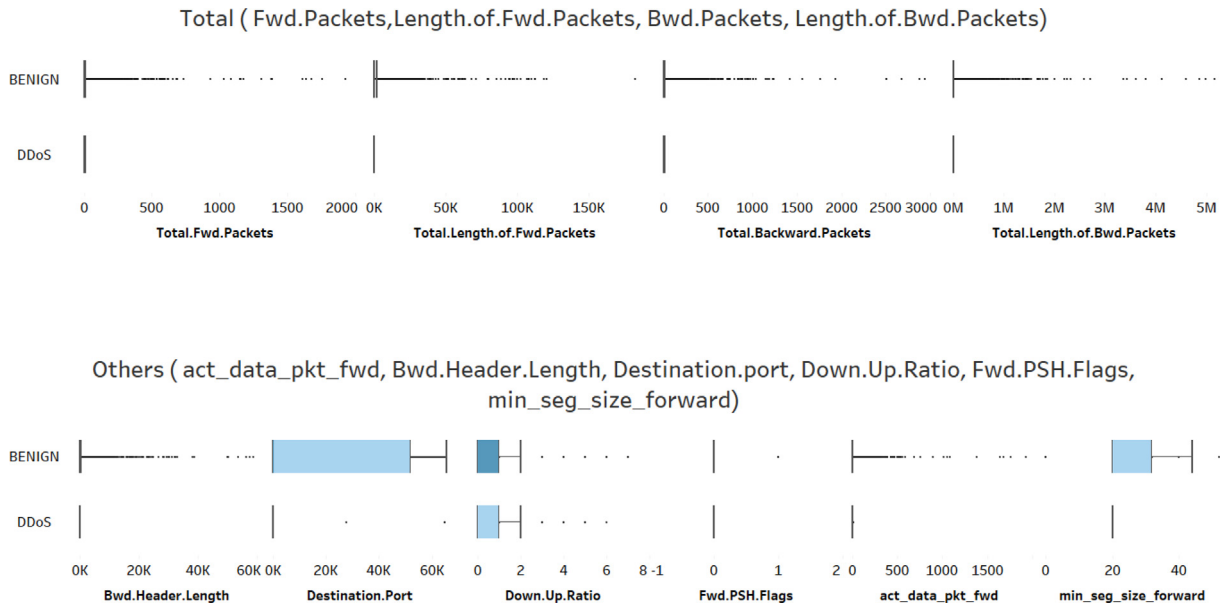

Fig. 16. Important attributes for (Total and Others) in visualization.

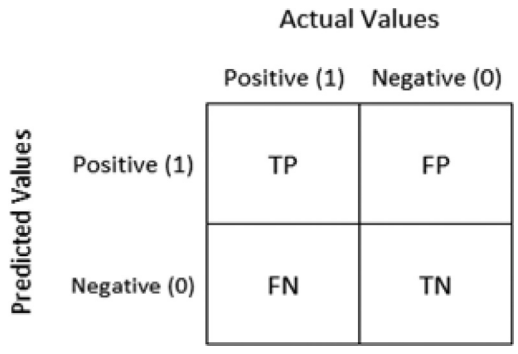

Fig. 17. A typical 2x2 Confusion Matrix.

of confusion matrix or contingency matrix are true positive (TP), false positive (FP), true negative (TN), false negative (FN), precision, recall, and F-measure. Fig. 17 shows the performance measure by using a typical confusion matrix.

TP = Number of correctly classified positive instances.
FN = Number of incorrectly classified negative.
FP = Number of incorrectly classified positive instances.
TN = Number of correctly classified negative instances.

This section started with the result of the Boruta algorithm for feature selection and followed with the result of classification using multiple classifiers (J48, random forest, naïve bayes, and multilayer perceptron).

Table 7 marked the performance measures (accuracy, F-measure, precision, and recall) of the multiple classifiers (J48, random forest, naïve bayes, and multilayer perceptron) in terms of effectiveness. It indicates that all the classifiers performed reason-

Table 7
Result of performance measures (accuracy, F-measure, precision, and recall) for multiple classifiers in cross-validation.

(%)	J48	Random Forest	Naïve Bayes	MLP (Multilayer Perceptron)
Accuracy	99.9848	99.993	97.826	99.104
Precision	100	100	97.8	99.1
Recall	100	100	97.7	99.1
F-measure	100	100	97.7	99.1
True Positive (TP) Rate	100	100	97.7	99.1
False Positive (FP) Rate	0	0	3	1

Table 8
Results of random forest in different number of iteration, accuracy and time consumption.

Maxdepth	Number Iteration	Accuracy	Time Consumption (Hours)	Maxdepth	Number Iteration	Accuracy	Time Consumption (Hours)
0[default]	**I = 100[default]**	99.993	3.5105	**2**	**I = 100**	98.7544	0.98
	I = 200	99.993	3.5105		**I = 200**	98.793	2.169
	I = 400	**99.994**	**7.117**		**I = 400**	98.7886	4.1765
	I = 600	99.9924	10.45		**I = 600**	98.7601	4.73
	I = 800	99.9924	18.3		**I = 800**	98.762	5.7625
	I = 1000	99.9924	15.33		**I = 1000**	98.7411	9.327
	I = 1500	99.9924	26.23		**I = 1500**	98.7335	11.79
	I = 2000	99.9918	35.45		**I = 2000**	98.7221	1.65
4	**I = 100**	99.9177	1.178	**6**	**I = 100**	99.9304	3.174
	I = 200	99.9196	2.854		**I = 200**	99.9297	3.914
	I = 400	99.9203	5.61		**I = 400**	99.9304	11.49
	I = 600	99.9196	8.7175		**I = 600**	99.9310	9.624
	I = 800	99.9209	10.925		**I = 800**	99.9310	13.442
	I = 1000	99.9209	13.129		**I = 1000**	99.9310	15.9575
	I = 1500	99.9209	24.721		**I = 1500**	99.9304	25.019
	I = 2000	99.9209	23.477		**I = 2000**	99.9304	32.051
8	**I = 100**	99.9684	1.507	**10**	**I = 100**	99.9804	1.872
	I = 200	99.9709	3.055		**I = 200**	99.9804	3.2625
	I = 400	99.9684	6.183		**I = 400**	99.9810	6.767
	I = 600	99.9684	9.198		**I = 600**	99.9816	9.012
	I = 800	99.9690	12.367		**I = 800**	99.9804	13.082
	I = 1000	99.9684	15.563		**I = 1000**	99.9804	16.4286
	I = 1500	99.9690	21.828		**I = 1500**	99.9804	6.585
	I = 2000	99.9684	31.767		**I = 2000**	99.9804	32.572

ably well, thereby boosting the result of the performance measures when using the Boruta as feature selection algorithm. The true positive rate (TPR) rate for the random forest (RF) was of the highest value, while the false positive rate (FPR) is only marked by naïve bayes only.

4.1. Evaluation tuning performance of Random forest algorithm.

The accuracy of the Random Forest algorithm was found to be highest among all the classifiers when detecting the DDoS. Hence, for this section, it was used as a base learner. Although it has several parameters, only two would influence the amount of pruning [12]. In Random Forest, the hyperparameters include the number of decision trees in the forest, and the number of features considered by each tree, when splitting a node. The parameters of Random Forest are the variables and thresholds used to split each node learned during training. In this experiment, number on iterations and maxdepth will be used to tune the performance of Random Forest so as to discover the progress of the accuracy in DDoS detection:

1. **number of iterations**: this determines the number of trees included in the ensemble. Each iteration yields a single tree. Increasing this value constructs the model more articulate which improves the accuracy of the training data. However, if set too high, the accuracy rate may diminish.
2. **max depth**: maxdepth refers to the maximum depth of each tree in the forest. The model becomes more expressive and effective even when the depth of the model is increased. Deep trees, on the other hand, are more difficult to train; they are capable of approximating. However, when employing Random Forest, it is acceptable to train deeper trees than when using a single decision tree. Overfitting is more frequent in a single tree than in Random Forest because of the variance reduction caused by averaging the multiple trees in the forest.

4.2. Tuning performance of Random forest algorithm

Table 8 tabulates the evaluation findings, with the highest score noted in bold. The highest evaluation result of Random Forest was an outstanding 99.994 %, in the parameter of maxdepth (0). The number of iterations were 400, and 600 in this table. This may be hypothesized as the lower the number of maxdepth, and iterations, the higher the accuracy. Due to the large volume of data, the process required a significant amount of time and energy consumption. This indicates that the higher the number of iterations used to fine-tune the performance, the longer it takes to achieve the accuracy.

Fig. 18 visualizes all accuracies marked with different number of iterations between 100 until 2000. Surprisingly, it shows that in iteration of maxdepth 2 parameter, the accuracies are low compared to others. This indicates that, this parameter is unsuitable for

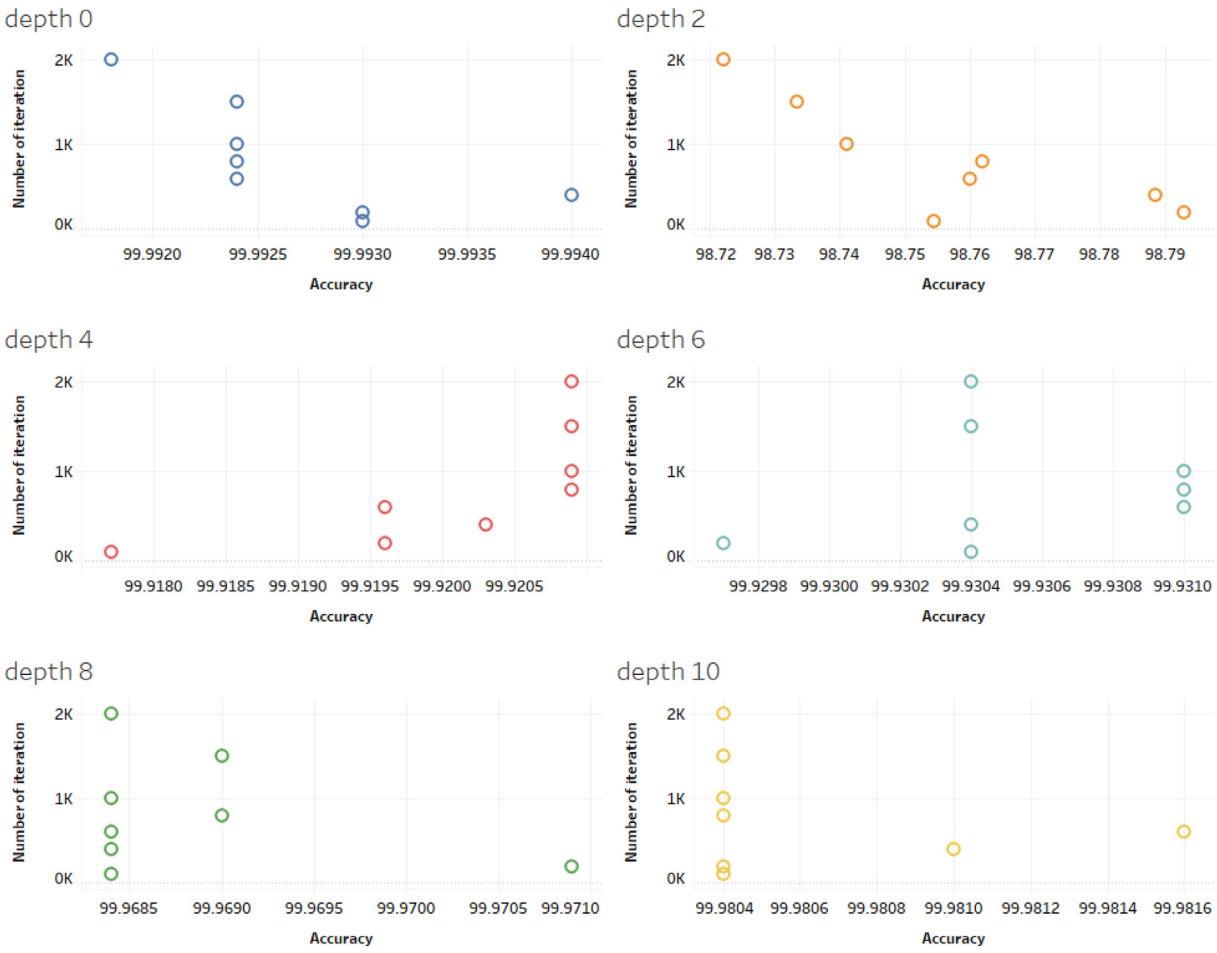

Fig. 18. Visualization of number of iteration vs accuracy.

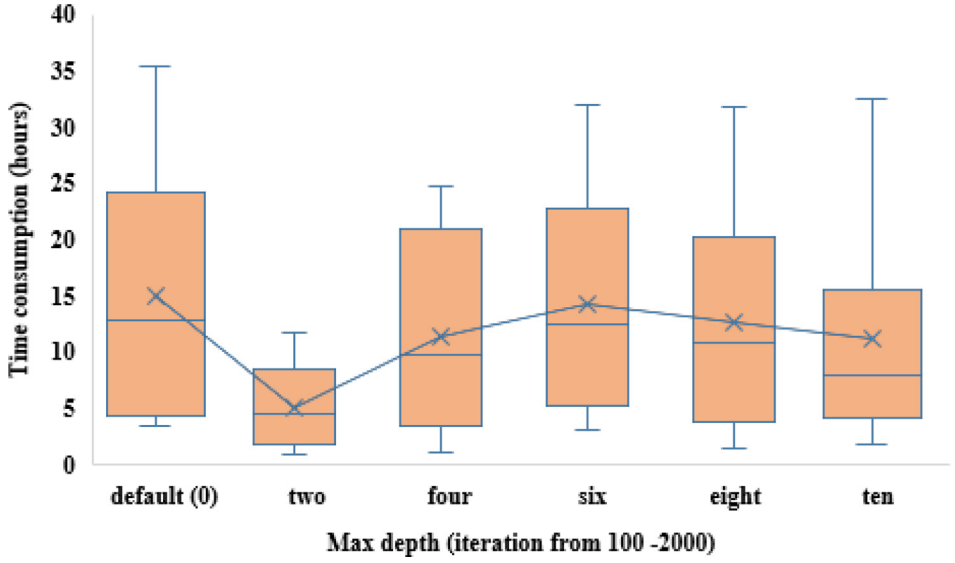

Fig. 19. The box plot of the time and maxdepth.

Table 9
Testing Boruta features in updated DDoS samples.

CICIDS2019 testing samples	Features	
	Boruta	Without Boruta
1	√	x
2	√	x
3	√	x
4	√	x
5	√	x
6	√	x
7	√	x
8	√	x
9	√	x
10	√	x

Legends:
√ Detected as DDoS.
x Undetected.

DDoS detection. Furthermore, the increment of the number of iterations, will not promise the increment of the accuracies. As depicted in depths of 0, 2, 6, 8 and 10.

4.3. Time consumption

In machine learning training processing time, it is crucial to consider the time consumption. This is imperative as machine learning will need to train and create a new model each time there is an update of new samples and features. Hence, in the interest to analyze the time consumption, Fig. 19 exhibits a box plot and shows that maxdepth of 2 provides the shortest time compared to others, followed by maxdepth 10. This discovers that maxdepth 2 only needs between 5 and 10 hours to train all the dataset, however, will marked the accuracies slightly less than other maxdepths. Nevertheless, maxdepth 10 is more suitable as it has better accuracies compared to maxdepth 2, and needs less time consumption compared to most of the other maxdepth parameters.

4.4. Testing model with CICIDS2019

In the previous section, this experiment uses DDoS dataset in CICIDS2017. Hence, in the interest to evaluate the Boruta features efficiency, this section evaluates the updated DDoS dataset in 2019, which is CICIDS2019 [51,52]. Furthermore, this experiment adopts Random Forest machine learning classifier for testing purposes. Table 9 tabulates the results and marks that Boruta features are able to detect the latest samples, compare to the training model that was constructed without Boruta. This proves that the machine learning model of Boruta features are able to detect unknown DDoS, compared to the training model without Boruta features.

5. Limitation and future studies

This study is constrained by a few limitations. Firstly, a large number of data entries from the CICIDS2017 dataset were used. It had 225,746 rows, and 79 columns, both of which could diminish the performance of machine learning. The existence of strings in the dataset, such as the 'Inf' string at 22 rows, and another certain column, could make the Random Forest classifier inoperative due to the algorithm supporting the statistical data. Therefore, we replace the 'Inf' with a suitable value.

On the other hand, using the Boruta algorithm for best feature selection may also be time consuming as the run time to complete this process took more than six hours for 99 iterations. Due to this deficiency, the data cannot be traced for every iteration. This is hampered by the large number of features present in the dataset. To solve this higher time consumption, there is a need to use a higher specification of computer for larger DDoS dataset experiments.

For future work, it may be possible to add more feature selection methods or algorithms after the Boruta algorithm. This additional approach would assist the Boruta algorithm and further reduce the list of features.

6. Conclusion

A Distributed Denial of Service (DDoS) is an attack that uses multiple distributed resources against the target. This study utilized the Boruta algorithm for feature selection by using the CICIDS2017 dataset. This dataset carries many features, comprising 79 features and 225,746 data network samples. The high volume of features tends to decrease the performance of machine learning. This study aimed to measure the ability of the Boruta algorithm to achieve the best feature for machine learning detection. In the machine learning training model process, it takes more than six hours to complete 99 iterations. The results showed that the number of features had decreased from 79 features to 65 features, including labels, eliminating a total of 14 features. After the best features had been selected, the data was split into two groups: train and test datasets (70 % and 30 %). These datasets were then used for the classification method and to develop a model with the Random Forest machine learning classifier. Random Forest offers an accuracy of 99.994 %, after tuning performance was done by using maxdepth and numIteration. Based on the tuning process and experiments conducted, it can be concluded that the Boruta algorithm provided better accuracy and could detect unknown DDoS attacks when compared to this study model with the other DDoS dataset in the future.

Funding

The authors gratefully acknowledge the Universiti Teknologi MARA (UiTM), Perak branch for financial support.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] O. Rahman, M. A. G. Quraishi, and C. H. Lung, "DDoS attacks detection and mitigation in SDN using machine learning," *Proceedings - 2019 IEEE World Congress on Services, SERVICES 2019*, vol. 2642–939X, pp. 184–189, 2019, doi: 10.1109/SERVICES.2019.00051.
[2] E. Ranjan, S., Swaminathan, R., Uysal, M., & Knightly, "DDoS-Resilient Scheduling to Counter Application Layer Attacks under Imperfect Detection," in *Proceedings IEEE INFOCOM. 25TH IEEE International Conference on Computer Communications*, 2006, pp. 1–13. [Online]. Available: https://doi.org/10.1109/INFOCOM.2006.127.
[3] A. Karimazad, R., & Faraahi, "An Anomaly-Based Method for DDoS Attacks Detection using RBF Neural Networks," in *International Conference on Network and Electronics Engineering* , 2011, vol. 11, pp. 44–48.
[4] D. Zargar, S. T., Joshi, J., & Tipper, "A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks," in *in IEEE Communications Surveys & Tutorials*, 2013, vol. 15(4), pp. 2046–2069. [Online]. Available: https://doi.org/10.1109/SURV.2013.031413.00127.
[5] A. Srivastava, B. B. Gupta, A. Tyagi, Anupama Sharma, and Anupama Mishra, "A recent survey on DDoS attacks and defense mechanisms," in *International Conference on Parallel Distributed Computing Technologies and Applications*, 2011, pp. 570–580.
[6] R. Suresh, M., & Anitha, "Evaluating Machine Learning Algorithms for Detecting DDoS Attacks," in *International Conference on Network Security and Applications*, 2011, vol. 196, pp. 441–452. [Online]. Available: https://doi.org/10.1007/978-3-642-22540-6_42.

[7] A. G. Amadi EC, Anakenyi D, Njoku C, "Study and Evaluation of Recent Ddos Trends of Attack on Web Server," *International Journal For Research In Advanced Computer Science And Engineering*, vol. 2(6), pp. 01–10.

[8] Kaspersky, Yaroslav Shmelev, Oleg Kupreev, and Alexander Gutnikov, "DDoS attacks in Q3 2021," *https://securelist.com/ddos-attacks-in-q3-2021/104796/*, Nov. 08, 2021.

[9] T. Matthews, "Incapsula Survey : What DDoS Attacks Really Cost Businesses," *Cyentia Cybersecurity Research Library*, 2017. http://lp.incapsula.com/rs/804-TEY-921/images/DDoS Report Q2 2015.pdf.

[10] F. Ullah, M. Edwards, R. Ramdhany, R. Chitchyan, M. A. Babar, and A. Rashid, "Data Exfiltration: A Review of External Attack Vectors and Countermeasures," *Journal of Network and Computer Applications*, vol. 101, no. October 2017, pp. 18–54, 2018, doi: https://doi.org/10.1016/j.jnca.2017.10.016.

[11] Feizollah A, Anuar NB, Salleh R, Wahab AWA. A Review on Feature Selection in Mobile Malware Detection. Digit Investig 2015;13:22–37.

[12] Blum AL, Langley P. Selection of relevant features and examples in machine learning. Artif Intell 1997;97(97):245–71. doi: https://doi.org/10.1016/S0004-3702(97)00063-5.

[13] Firdaus A et al. Selecting root exploit features using flying animal-inspired decision. Indones J Electr Eng Informat 2019;7(4):628–38. doi: https://doi.org/10.11591/ijeei.v7i4.1146.

[14] Jusoh R, Firdaus A, Anwar S, Osman MZ, Darmawan MF, Ab Razak MF. Malware detection using static analysis in Android: a review of FeCO (features, classification, and obfuscation). PeerJ Comput Sci 2021;7(e522):1–54. doi: https://doi.org/10.7717/peerj-cs.522.

[15] Mat SRT, Razak MFA, Kahar MNM. Juliza Mohamad Arif, Salwana Mohamad, and Ahmad Firdaus, "Towards a systematic description of the field using bibliometric analysis: malware evolution". Scientometrics Feb. 2021;126:2013–55.

[16] A. Feizollah, A., Anuar, N. B., Salleh, R., Wahid, A., & Wahab, "A review on feature selection in mobile malware detection," *Digit Investig*, vol. 13, pp. 22–37, 2015.

[17] D.: Song, F., Guo, Z., Mei, "Feature selection using principal component analysis," *In: 2010 International Conference On System Science, Engineering Design and Manufacturing Informatization (ICSEM)*, vol. 1, pp. 27–30, 2010.

[18] Patil NV, Krishna CR, Kumar K. SSK-DDoS: distributed stream processing framework based classification system for DDoS attacks. Cluster Comput 2022;25(2):1355–72. doi: https://doi.org/10.1007/s10586-022-03538-x.

[19] Patil NV, Rama Krishna C, Kumar K. S-DDoS: Apache spark based real-time DDoS detection system. J Intell Fuzzy Syst 2020;38(5):6527–35. doi: https://doi.org/10.3233/JIFS-179733.

[20] Wani S, Imthiyas M, Almohamedh H, Alhamed KM, Almotairi S, Gulzar Y. Distributed denial of service (Ddos) mitigation using blockchain—a comprehensive insight. Symmetry (Basel) 2021;13(2):1–21. doi: https://doi.org/10.3390/sym13020227.

[21] R. B. He, Z., Zhang, T., & Lee, "Machine Learning Based DDoS Attack Detection from Source Side in Cloud," *Proceedings - 4th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud and 3rd IEEE International Conference of Scalable and Smart Cloud, SSC*, pp. 114–120, 2017, [Online]. Available: https://doi.org/10.1109/CSCloud.2017.58.

[22] S. Sanmorino, A., & Yazid, "DDoS Attack Detection Method and Mitigation Using Pattern of the Flow," *2013 International Conference of Information and Communication Technology, ICoICT 2013*, pp. 12–16, 2013.

[23] Maniam JN, Singh D. Towards Data Privacy And Security Framework In Big Data Governance. Int J Software Eng Comput Syst (IJISCT) May 2020;6(1):41–51. doi: https://doi.org/10.15282/ijsecs.6.1.2020.5.0068.

[24] A. Gupta, "Distributed Denial of Service Attack Detection Using a Machine Learning Approach," 2018. Accessed: Oct. 03, 2022. [Online]. Available: https://prism.ucalgary.ca/bitstream/handle/1880/107615/ucalgary_2018_gupta_animesh.pdf?sequence=3&isAllowed=y.

[25] Karim A, Chang V, Firdaus A. Android botnets: A proof-of-concept using hybrid analysis approach. J Organizat End User Comput 2020;32(3):50–67. doi: https://doi.org/10.4018/JOEUC.2020070105.

[26] S. Sahu, A. Verma, "DDoS attack detection in ISP domain using machine learning," in *International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 2019, vol. m, pp. 1–4.

[27] Y. Zekri, M., Kafhali, S. El, Aboutabit, N., & Saadi, "DDoS Attack Detection using Machine Learning Techniques in Cloud Computing Environments," *Proceedings of 2017 International Conference of Cloud Computing Technologies and Applications, CloudTech 2017*, pp. 1–7, 2018, [Online]. Available: https://doi.org/10.1109/CloudTech.2017.8284731.

[28] Nguyen Y, Choi HV. Proactive detection of DDoS attacks utilizing k-NN classifier in an AntiDDoS framework. Int J Electr Comput Syst Eng 2010.

[29] Usoro A. Computing and Information Systems Journal. Comput Informat Syst J 2016;20(1):1–31.

[30] Rawashdeh; Adnan & Al-kasassbeh; Mouhammd & Al-Hawawreh, *An anomaly-based approach for DDoS attack detection in cloud environment*. 2018. doi: 10.1504/IJCAT.2018.093533.

[31] P. Satam, "Anomaly Based Wi-Fi Intrusion Detection System," *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2017, doi: 10.1109/fas-w.2017.180.

[32] Ghous H, Malik MH, Abbas M, Ismail M. Early Detection of Breast Cancer Tumors using Linear Discriminant Analysis Feature Selection with Different Machine Learning Classification Methods. Int J Informat Syst Comput Technol (IJISCT) 2022;1(1):1–12. doi: https://doi.org/10.5121/cseij.2022.12117.

[33] Sajjad M, Pasha M, Pasha U. Parametric Evaluation of E-Health Systems. Int J Informat Syst Comput Technol (IJISCT) 2022;1(January):31–7.

[34] Alkasassbeh M, Hassanat ABA, Al-naymat G. Detecting Distributed Denial of Service Attacks Using Data Mining Techniques. Int J Adv Comput Sci Appl 2016;7(1):436–45.

[35] R. Tang and X. Zhang, "CART Decision Tree Combined with Boruta Feature Selection for Medical Data Classification," in *5th IEEE International Conference on Big Data Analytics (ICBDA)*, 2020, pp. 80–84.

[36] Kursa MB. Feature Selection with the Boruta Package. J Stat Softw 2010;36(11):1–13.

[37] Adam Kordeczka, "Boruta - modern dimension reduction algorithm," 2018. [Online]. Available: http://rstudio-pubs-static.s3.amazonaws.com/369273_87ccc31e36c44bb886a5dfbf5865bb1c.html.

[38] S. S. Priya, M. Sivaram, D. Yuvaraj, and A. Jayanthiladevi, "Machine Learning based DDOS Detection," in *International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2020, pp. 234–237.

[39] W. T. Aung; Y. Myanma ; K. H. M. S. Hla, "Random forest classifier for multi-category classification of web pages," *Proceeding of the IEEE Asia-Pacific Conference on Services Computing, Singapore, Singapore, 7–11 December, 2009*, 2009.

[40] Kotsiantis S. Integrating global and local application of naive Bayes classifier. Int Arab J Inform Technol 2014;11:300–7.

[41] S. Das, A. M. Mahfouz, D. Venugopal, S. Shiva, "DDoS Intrusion Detection through Machine Learning Ensemble," *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 471–477, 2019, doi: 10.1109/QRS-C.2019.00090.

[42] S. Shabbir, "Comparing Performance of J48 , Multilayer Perceptron (MLP) & Naïve Bayes (NB) Classifiers on Breast Cancer Data Set using WEKA," no. October, 2018, doi: 10.13140/RG.2.2.30639.79522.

[43] Alam F, Pachauri S. Comparative Study of J48, Naive Bayes and One-R Classification Technique for Credit Card Fraud Detection using WEKA. Adv Computat Sci Technol 2017;10(6):1731–43.

[44] S. Shabbir, "Comparing Performance of J48 , Multilayer Perceptron (MLP) & Naïve Bayes (NB) Classifiers on Breast Cancer Data Set using WEKA," in *CS Graduate Student at NEIU Northeastern Illinois University (NEIU)*, 2018, no. October. doi: 10.13140/RG.2.2.30639.79522.

[45] Rabie MT, Nassif AB, Alaaeddin M. Regression Analysis of Solar Flares: A Multilayer Perceptron Approach with Feature Selection Techniques. Int J Comput Commun 2020;14:84–90. doi: https://doi.org/10.46300/91013.2020.14.14.

[46] R. F. Fouladi, C. E. Kayatas, E. Anarim, "Frequency based DDoS attack detection approach using naive bayes classification," *2016 39th International Conference on Telecommunications and Signal Processing, TSP 2016*, pp. 104–107, 2016, doi: 10.1109/TSP.2016.7760838.

[47] C. Akmal Che Yahaya, C. Yahaya Yaakub, A. Firdaus Zainal Abidin, M. Faizal Ab Razak, N. Fatin Hasbullah, and M. Fadli Zolkipli, "The prediction of undergraduate student performance in chemistry course using multilayer perceptron," *IOP Conf Ser Mater Sci Eng*, vol. 769, p. 012027, 2020, doi: 10.1088/1757-899x/769/1/012027.

[48] A. A. Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *In Proceedings of the 4th International Conference on Information Systems Security and Privacy* , 2018, pp. 108–116. [Online]. Available: https://doi.org/10.5220/0006639801080116.

[49] Gniewkowski M. An Overview of DoS and DDoS Attack Detection Techniques. International Conference on Dependability and Complex Systems. Cham: Springer; 2020.

[50] Sidana M. Intro to types of classification algorithms in Machine Learning. Sifium 2017. , https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e1.

[51] I. Sharafaldin, A. H. Lashkari, S. Hakak, A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–8. doi: 10.1109/CCST.2019.8888419.

[52] C. I. for Cybersecurity, "DDoS Evaluation Dataset (CIC-DDoS2019)." https://www.unb.ca/cic/datasets/ddos-2019.html (accessed Apr. 23, 2022).