# Influential Features Analysis And AI-Driven Accuracy Enhancement: A Study Case For DDoS Detection

*Le Ba Nguyen*[1], *Quoc-Binh Nguyen*[2], *Ngoc Hong Tran*[1,*]

[1]Computer Science Program, Vietnamese-German University, Binh Duong Province, Viet Nam.
[2]Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Viet Nam.

*Corresponding Author: Ngoc Hong Tran (email:ngoc.th@vgu.edu.vn)
(Received: 05-June-2024; accepted: 14-October-2024; published: 31-December-2024)
http://dx.doi.org/10.55579/jaec.202484.466

***Abstract.*** *Cybersecurity is known today as one of the greatest challenges of the modern era. Among the various types of cyber attacks that threaten our security, the Distributed Denial of Service (DDoS) attack is among some of the most common, effective, and well-recognized attack strategies. Since this form of attack is meant to disrupt the availability factor covertly, it can be detrimental to the targeted machines and difficult to be discovered. Because of that, there have been a number of approaches, as well as solutions that have been devised in order to detect it as accurately and efficiently as possible. Impressively, data mining methods have been employed to identify patterns of DDoS attacks from the computer network traffic. Nevertheless, the recent works' results have not yet mentioned which factors of the computer network traffic play the most vital role in indicating the potential for true positive attacks. Additionally, with the Machine Learning approach, there are still ample opportunities to enhance the attack prediction accuracy of the detection model. As such, in this paper, we attempt to explore factors that would influence the classification result, and leverage a variety of Machine Learning algorithms, i.e. Random Forest, Naive Bayes, Logistic Regression, and Multilayer Perceptron, for the purpose of improving the accuracy of data classification. The experiments were deployed using CICIDS2017 dataset and compared with the other related works on the same dataset. The experimental outcomes of our methodologies and analyses demonstrate some potential and effectiveness enhancement compared to previous works. Moreover, we analysized and concluded the insight of how side factors affect the attack identification result. The collected information from our analysis identifies dominant factors, and opens a new view for their hidden correlationship directly affecting the attack labeling.*

***Keywords:*** *Distributed Denial of Service, Random Forest, Naive Bayes, Logistic Regression, Multilayer Perceptron.*

## 1. Introduction

As of April 2023, the number of Internet users has reached over five billion, according to [1], which means that the number of computers and Internet connected systems is significant, which leaves plenty of rooms for security breaches. The key objectives of Internet security are authentication, integrity, availability, confidentiality, and non-repudiation [2]. Cyber attacks, particularly the Distributed Denial of Service (DDoS) attack, pose a serious threat to the availability aspect of

network security by flooding a target network or system with traffic from multiple sources, overloading its capacity in order to deny access for legitimate users [3]. The consequences of DDoS attacks can be devastating, leading to network downtime, loss of revenue, damage to reputation, cost for damage mitigation, etc. It can last anywhere from a few hours to days, or even weeks. Consequently, during an attack the service quality would attenuate, which could be financially damaging to businesses. Depending on the type and size of companies, a successful DDoS attack could cause a loss from around 100,000 to tens of millions of dollars per hour [4]. To cope with such egregious effect from DDoS attacks, the effective detection methods, and Intrusion Detection Systems (IDS) for such attacks are crucially requested. Under such a motivation, researches on applying Data Mining techniques to detecting DDoS attack have been remarkably invested, with approaches such as clustering, classification, association rule, and other techniques being used to analyze network traffic data in order to gain insight and handle the matter [5].

In [6], Konstantin Borisenko et al. conducted a study on using data mining techniques for detecting internal and external DDoS attacks on a cloud computing platform. They used a combination of live connection data and modeled attack traffic, focusing on HTTP and SYN flooding attacks. Their approach employed two layers of detection: the first layer identified attack traffic within the cloud network, while the second layer identified victims and attackers. They tested Naive Bayes, SVM, KNN, and Decision Tree models for the first layer, with Decision Tree proving to be the most effective. The first layer achieved a 0% false positive rate and a 0.03% false negative rate, correctly identifying all benign traffic. Despite some HTTP flooding passing through, its impact was negligible. Upon detecting an attack, the second layer initiated, successfully distinguishing victim and attacker nodes with near 100% accuracy across five steps. While the results were promising, the authors noted a potential 25% error rate reduction with binary classification.

In [7] , Rohan Doshi et al. applied a selected number of machine learning techniques, which are K-nearest neighbors (KNN), RF, Decision Tree, Support Vector Machines (SVM), and Deep Neural Networks (DNN), to detect DDoS over a simulated Internet of Things (IoT) device network. The collected data traffic first went through feature engineering before the classification process. Based on the outcomes of the classification algorithms, it was noted that the classifiers scored between 0.91 to 0.99 in accuracy, with SVM having the lowest accuracy and RF having the highest score.

In the most similar works, specifically in [8,9], the authors use the CICIDS2017 dataset for their machine learning (ML) algorithms to detect DDoS attack. In their work they used Random Forest (RF), Naive Bayes (NB), Support Vector Machine (SVM), and Decision Tree algorithms. The results and findings in their study are very positive and interesting. However, they did not provide more details about their data processing to support the ML algorithms, and the accuracy of the employed models could be further improved upon with different setup, and data processing approaches. In addition, their works did not explore much on which factors of the dataset has the greatest impact on the possibility of a traffic to be a DDoS attack.

Additionally, the work conducted in [10] used the same dataset, however, their goal was to apply and evaluate different ML techniques that could classify between the traffic flows that are normal and those that correspond to the different attacks present in the CICIDS2017 dataset. This is slightly different from our work in that, we only focus on the DDoS attack. Despite not sharing the same objective, their findings were still quite promising, and since we shared the same dataset, as well as many of the applied ML algorithms used, NB, LR, MLP, and RF, so we shall include it for the comparison of our findings (see Section 5).

In this work, we attempt to enhance the DDoS attack detection accuracy by refining the CICIDS2017 dataset. When we have a clean dataset, we employ various ML approaches, specifically, Random Forest (RF), Multilayer Perceptron (MLP), Naive Bayes (including Multinomial and Gaussian), and Logistic Regression (LR). Beyond that, we shall try

 

to address the key features that can suggest the presence of a potentially true positive DDoS attack on the computer network traffic. For confirming our result, we shall then get the results in [8,9] for a comparison.

The remainder of this paper is organized into 6 sections. Section 2. presents some related works to DDoS detection and this paper. In section 3. , an overview of DDoS and the deployed ML algorithms are demonstrated. Section 4. describes the dataset, as well as the processing method of it. Following up in section 5. , we discuss the results of our models and compare our findings with the most similar works. Finally, we give our concluding remarks and potential future research avenues in section 6. .

## 2. Related works

The threat of security breech, specifically those caused by DDoS attacks, and the growing population of Internet users has lead to many research being conducted on creating effective detection techniques and datasets to accompany them. In this section, we presents some contemporary works, as well as some available IDS datasets.

### 2.1. IDS datasets

#### 1) KDD99

Created by the University of California in 1999, it is an improved version of DARPA 98 and is created using tcpdump. It is comprised of approximately four gigabytes worth of compressed TCP data that is collected from seven weeks of simulated network traffic, with normal and attack traffic [11]. The dataset has 4 main categories of attack, those are Denial of Service (DOS), Remote to Local (R2L), User to Root (U2R), and probing. Although it is no longer up-to-date, as well as having a number of redundant records, it is still widely studied and appears in recent works, such as in [12] and [13].

#### 2) NSL-KDD

This dataset was created as a way to address some of the inherent problems in the KDD99 dataset [14]. The dataset consists of different features and labeled traffic to differentiate between normal and the different types of attack traffic. The attack categories are similar to that of KDD99 dataset, featuring Denial of Service (DOS), Remote to Local (R2L), User to Root (U2R), and probing. Many works have used it as a benchmark to evaluate the performance of their IDS, such as in [15].

#### 3) CAIDA-KDD

Center of Applied Internet Data Analysis (CAIDA) possesses various types of dataset, with three main category being ongoing, one-time snapshot, and complete [16]. Data is gathered by CAIDA from various locations, and each dataset has unique features such as UDP probing, BGP monitoring, IPv4 census with passive traffic traces obtained from an academic ISP, a darknet, and a residential BGP with active measurements of ICMP ping, HTTP GET, and traceroutes. The majority of databases have their payload and IP addresses anonymize, which significantly diminishes their utility [17].

#### 4) CICIDS2017

This dataset was created in 2017 by the Canadian Institute for Cybersecurity at the University of Brunswick [18]. The dataset, tools, as well as the codes used for its creation are publicly available. It contains 80 features for each data record, various types of attacks, which includes Brute Force, Heartbleed, different flavors of DoS and DDoS attacks, Web Attack, Infiltration, and Botnet. In addition, the dataset contains a realistic background traffic generated based on several protocols, such as HTTP, HTTPS, FTP, SSH, and SMTP [19]. Lastly, the dataset is in CSV file format, making it more convenience when using it for machine learning.

## 2.2. Previous works

Previous research in [20] by Mouhammd Alkasassbeh et al. used Data Mining techniques to detect DDoS attacks. The dataset used in their experiment covers Smurf, UDP Flood, HTTP-Flood, and SQL Injection DDoS attacks. MLP, Naive Bayes, and RF were among the algorithms used, their performance was assessed based on accuracy, precision, and recall score. The results indicated that MLP had the best overall outcome and the algorithms had difficulties in classifying the Smurf attack.

A study by Kimmi Kumari and M. Mrunalini [21] saw the use of a mathematical model and two ML algorithms, LR and Naive Bayes, for detecting DDoS attacks. In their experiment, they used the CAIDA 2007 dataset. According to their findings, the ML model was more accurate than the mathematical model, with 100% accuracy compared to 99.75% accuracy for the mathematical model. Because Naive Bayes assumed that dataset features are independent, which is not true in most circumstances, LR outperformed Naive Bayes.

In [9], Amer A. Abdulrahman and Mahmood K. Ibrahem presented a work that evaluates DDoS attack detection using ML algorithms. Their work classified benign and malicious DDoS traffic in the CICIDS2017 dataset using four ML models: C5.0, RF, Naive Bayes, and SVM. The results indicated that C5.0 and RF outperform other algorithms, they have the highest accuracy at 86.45% and 86.8%, respectively. They found that algorithm complexity depended on characteristics and training data samples. More features mean that more training data is needed.

The work in [22] by Muhammad Azmi and his team aimed to detect DDoS attacks using various classification algorithms and feature selection methods. They utilized Information Gain and Data Reduction for feature selection and applied Naive Bayes, Artificial Neural Network (ANN), and Decision Table algorithms on the UNSW-NB 15 dataset. The performance of the algorithms were evaluated based on accuracy, precision, true positive and false positive rate. Results revealed that Decision Table with the highest accuracy at 88.43%, followed by Naive Bayes at 87.74%, and ANN at 84.66%. Feature selection enhanced the accuracy of Naive Bayes by 5.77%, Decision Table by 3.89%, and ANN by 0.68%.

In [23], Priyanka Kamboj et al. conducted a survey to review how different methods could help with detecting DDoS attacks. In their work, they discuss about types of DDoS attack that could be used on targeted victims, such as SYN Flooding, ICMP Flooding, UDP Flooding. The study also discuss on some existing detection methods that could be deployed as a part of counter measures against DDoS, those are Traceback Methods, Entropy Variation, Intrusion Detection and Prevention Systems. The research found that DDoS attacks are complicated issues, with varying degrees of difficulty, and that each detection methods have their own strength and weaknesses.

M Devendra Prasad et al. introduced a Stochastic Gradient Boosting (SGB) ML model for DDoS anomaly detection in [24]. The model is trained using a dataset generated from three different datasets. From the result, SGB fared better than K Nearest Neighbors (KNN), Naive Bayes, Decision Tree, and RF. It achieved 100% accuracy with no false classifications for balanced and unbalanced datasets. They concluded that by adapting SGB, a system for DDoS detection can be fully automated obviating human intervention. In [8], Saman Sarraf successfully replicated the work of Prasad and his colleagues in [11]. For his work, he used a fraction of the CICIDS2017 dataset, while his work was aiming to replicate used the complete dataset in combination with two other datasets. The study showed that both models had an accuracy rate of almost 100% and it was possible to replicate the results of the original research.

In [10], Maria Rodriguez et al. evaluated the accuracy and execution time of ten ML algorithms for classifying traffic flows in the CICIDS2017 dataset. For their study, they applied Naive Bayes, LR, MLP, Sequential Minimal Optimization (SMO), KNN, Adaptive Boosting, OneR, J48, PART, and RF. The models went through three rounds of training and testing, first with the original dataset, the second had

 

feature selection, the last used a dataset generated by Zeek network traffic analysis on raw packet captures. They noted that binary classification achieved better results due to the reduced complexity, it need less time to test if only the most relevant data attributes were used, but at the cost of accuracy. Additionally, tree-based algorithms, particularly PART and J48, outperformed others, consistently achieving over 99% accuracy across all trials and proving to be faster alternatives to RF.

Receiving such great inspiration, in this work, we deploy the CICID2017 dataset for our study due to its availability to the public and comprehensive nature, realistic network traffic scenarios, and extensive feature set allowing for in-depth analysis and evaluation of detection algorithms. We are only considering the studies in [8, 9] are the most similar to ours so far because we share the same CICIDS2017 dataset. To improve prediction accuracy, we use several ML algorithms and data processing that are distinct from them.

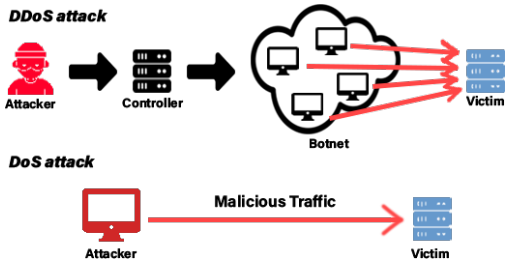


**Fig. 1:** DDoS and DoS attack illustration.

# 3. Preliminary

## 3.1. DDoS attack

DDoS is a variant of the Denial of Service (DoS) attack, which is a type of cyber attack that seeks to make a machine or network resource unavailable to its intended users by interrupting the device's normal functions [2], typically by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, denying access to additional users (see Fig. 1). DoS attack is characterized by using a single computer to launch the attack, while in DDoS the attack originates from numerous sources [3] (see Fig. 1).

The attacks are executed through the use of a network of malware-compromised devices that could be remotely controlled by an attacker. Infected machines are known as bots or zombies, and a collection of them is called a botnet. After the botnet is established, the attacker can launch an attack on a victim's server or network by issuing instructions to each bot to overwhelm the victim with requests.

## 3.2. Machine learning Methods

In this section, we introduce the ML approaches applied to our problem. We elected to employ these models for this study because our objective is binary classification in nature and they are known by many to be some of the most suitable, as well as widely utilized techniques.

### 1) Logistic Regression (LR)

A type of statistical method for binary classification that can also be applied to linear classification [25]. It calculates the likelihood of an event, such as "True" or "False", using a dataset of independent variables. Since the result is a probability, the dependent variable is limited between 0 and 1. LR transforms odds, the ratio of success to failure, into a logit function.

$$P = \frac{1}{1 + e^{-(a+bX)}} \tag{1}$$

where $P$ is the probability of a result we need; $e$ is the base of the natural logarithm (or Euler's number); $a$ and $b$ are the parameters of the model; $X$ is the independent variable.

### 2) Random Forest (RF)

This popular model is used to solve both classification and regression problems. It takes the results of multiple decision trees and through ensemble learning combines them into a single result [25]. It's the same as a group of people voting for an outcome based on their own individual opinions (see Fig. 2). RF leverages clas-

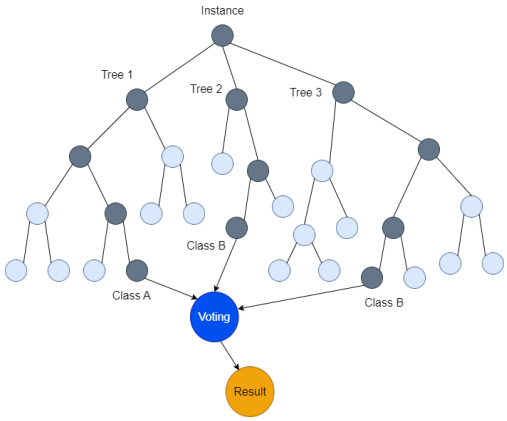


**Fig. 2:** Random Forest illustration.

sifier accuracy and dependency to establish an upper bound on generalization error. Reducing correlation can enhance its strength and accuracy. RF involves three key parameters, those are node dimension, tree quantity, and sampled feature count. Each decision tree in the forest starts with a root node and gradually splits between a criteria until it arrives at a conclusion at a leaf node. The quality of the split is judged by metrics such as *Gini* index.

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

where $S$ is the set with n classes; $p_i$ the probability of randomly picking an element of class $i$ from the set $S$.

### 3) Naive Bayes

The Naive Bayes classifier is a supervised ML algorithm, it is a fast and simple classification algorithm that makes it suitable for high-dimensional datasets [26]. It uses Bayes' theorem (also known as Bayes' rule) and a firm assumption that the attributes are conditionally independent, given the class.

$$P\,(A|B) = \frac{P\,(A)\,P\,(B|A)}{P\,(B)} \tag{3}$$

where $P\,(A|B)$ is the probability of event $A$ occurring given that event $B$ has occurred; $P\,(B|A)$ is the probability of event B occurring given that event A has occurred; $P\,(A)$ is the probability of event A; $P\,(A)$ is the probability of event $B$.

Despite that in practice independence condition is uncommon, Naive Bayes usually yields competitive classification accuracy [27]. In the Gaussian Naive Bayes (GNB) classifier the model assumes that data from each label is drawn from a simple Gaussian distribution [26]. Another type of Naive Bayes is Multinomial Naive Bayes (MNB), where the features are assumed to be generated from a simple Multinomial distribution [26].

### 4) Multilayer Perceptron (MLP)

A type of artificial neural network characterized by its fully connected feed-forward architecture [25]. It consists of a minimum of three layers, namely the input layer, the output layer, and at least one hidden layer. The MLP architecture consists of interconnected layers of input nodes, also known as neurons, forming a directed graph structure, wherein edges have specific directional connections from the input layer out to the output layers (see Fig. 3).

Hidden layers consist of all of the layers that lie from after the input layer to before the output layers [28]. The application of back-propagation is suitable for training both MLPs as well as deep neural networks, which can be considered as multilayered MLPs. The output of a neuron in the network is typically computed using the equation in Eq. 4.

$$y = f \times \left( \sum_{i=1}^{n} w_i \times x_i + b \right) \tag{4}$$

where $y$ is the output of the neuron; $f$ is the activation function; $w_i$ is the weight of the connection between the neuron and its inputs; $x_i$ is the input value; $b$ is the bias term.

We selected these models based on the model effectiveness in binary classification task, their ability to handle both linear and nonlinear relationships, and their interpretability and computational efficiency. Some models, such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN), were considered but excluded from the study due to their slower performance and lower effectiveness found in other works.

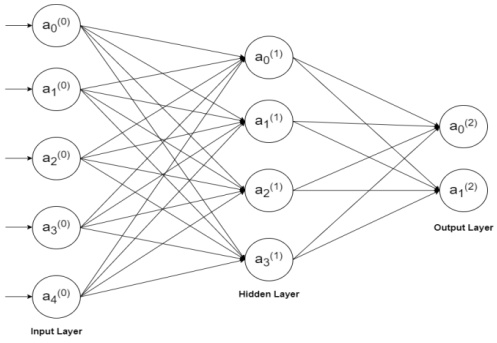


**Fig. 3:** Multilayer Perceptron illustration.

# 4. Dataset processing

This section provides an overview of the dataset that was used for this study, as well as the utilized approach in processing of the data before it could be used to train and test the ML models.

## 4.1. Dataset description

For this work, the CICIDS2017 dataset was utilized due to it containing bidirectional traffic flows with benign traffic and different types of up-to-date attacks, it is also a widely used dataset, in addition to it being publicly accessible. Other datasets were considered but they were not as accessible and they didn't have as many features as CICIDS2017. The dataset was created by the Canadian Institute for Cybersecurity (CIC) and is publicly accessible on their website. The dataset imitates real-world data traffic and contains common cyber attack types [18] that were recorded over the course of five days by the CIC, with each day having a different type of attack, except for Monday that only has Benign data. Since our interest lies in DDoS, we only used the file created on Friday named "Friday-WorkingHours-Afternoon-DDos" that contains this type of attack.

The dataset comprises of 85 features and a total of 225,745 instances, each associated with one of two class labels "Benign" or "DDoS". The features encompass labeled flows containing pertinent details such as time stamps, IP addresses, ports, protocols, and other elements that closely resemble authentic network traffic data. It has 128,027 occurrences of "DDoS" and 97,718 instances of "Benign", which is around 1.31:1 ratio, a mild imbalance which does not impact much on the result. Therefore, the data consists of around 43% "Benign" instances and 57% "DDoS" cases.

There are analytical data of network traffic created from CICFlowMeter, which is a tool created by the CIC for generating and looking into network traffic flow. The source code of the tool is open to the public and can be accessed on their website [29] and it possesses the capacity to create a maximum of 84 attributes that are linked to bidirectional flows. However, there exist a number of flaws within the dataset. In a study by Arnaud Rosay and his team [30], there are some issues with the tool used for the creation of the dataset, as well as the dataset itself. These issues is likely to cause a reduction to the system performance. Further details of the reported issues are discussed in section B.

## 4.2. Existing challenges

The dataset contains a number of challenges that need to be address before it can be used to train the models.

### 1) Unclear Version and Inability to Replicate Data

The CICFlowMeter tool has received changes over time, however, the absence of a definitive version tag in its source code has rendered it challenging to determine the precise version required for replicating the extracted dataset features, as well as features that are in earlier versions of the tool but is unavailable in the public access version of the source code.

### 2) Feature Duplication

Some of the features were detected to be redundant within the dataset due to it being a copy of other features. The first pair is "Average Packet Size" and "Packet Length Mean", because they both use the same average length of packets

per flow values, so they are duplicated features. Next, the feature "Fwd Packet Length Mean" is found to be overlapped with "Fwd Segment Size Avg". A similar thing happens when you look at the features "Fwd Header Length" and “Fwd Header Length.1” features, they have identical names and values. "Bwd Packet Length Mean" and "Bwd Segment Size Avg" are another pair of features that are duplicated.

### 3) Feature Miscalculation

Several features within the dataset were found to be flawed due to calculation mistakes. There are around 23 flaws in the feature calculation; if we consider only the non-duplicated features, then we have 21 miscalculations left.

- The most common flaw is the calculation errors, such as dividing integers when they should be diving floating point values. This mistake affected 6 bulks related features, 4 related to subflows and 1 to downlink/uplink ratio.

- Each new packet updates backward bulk features regardless of packet direction. Also, the forward bulk features never get any updates. This leads to all 6 of the bulk-related features having false values.

- Subflow count gets updated by timestamp testing values. Misplaced parenthesis make the test always true, raising subflow count for each packet received.

- Each flow's first packet gets used twice for updating features related to packet length, which affects the standard deviation, total packet length, and mean values.

- There are flaws in TCP-related features, the counts for SYN, PSH, FIN, and URG flags are all reversed. PSH and URG counts never get updated for any direction. Every time a packet is received, the backward initial TCP window size is changed to the most recent number instead of the first one.

### 4) Wrong protocol detection

CICFlowMeter processes the packets in line with the identified protocol. Regretfully, there are times when the tool is unable to identify it. Inadequate analysis of the Ethernet frames results in different kinds of protocol detection issues.

- The initial detection level in the CICFlowMeter tool relies on inspecting whether a packet contains an IPv4 or IPv6 header, rather than the Ethertype field in the Ethernet header as intended. Regardless of the Ethertype field, packets are classified as IPv4 or IPv6 if their payload resembles these headers. This leads to misclassification of Address Resolution Protocol (ARP), Link Layer Discovery Protocol (LLDP), and Cisco Discovery Protocol (CDP) packets, as observed in a PCAP file analysis. For instance, some ARP packets are misinterpreted as IPv4 packets due to matching IP addresses with ARP header fields.

- CICFlowMeter's second analysis level, intended to rely on the protocol field of the IPv4 header, often misidentifies packets containing TCP or UDP headers due to potential confusion with payload bytes. This leads to misclassification of various protocols such as Stream Control Transmission Protocol (SCTP), Internet Group Management Protocol (IGMP), and Internet Control Message Protocol (ICMP), with most packets being associated with protocol "0". Additionally, there are observed instances of broken frames, particularly with UDP fragments not being correctly identified and categorized.

### 5) Inconsistent TCP Termination Flows

The CICFlowMeter tool ends connection flows either after 120 seconds or upon encountering a 'FIN' flag in a packet. In a TCP connection, the first flow is established when a connection is made, which includes the 3-way handshake and data transfer packets. Closure of this flow occurs upon receiving the initial packet of the last 4-way handshake. The subsequent flow includes the second and third packets of this handshake. The final packet initiates a third flow. The tool used to create the dataset left some issues.

- If there is no other communication that utilizes the exact same addresses or ports then the third flow will timeout and close. Else, the last

 

flow packet will be added to a new communication.

- Flow-based features like 4-way handshakes during normal and attack traffic can significantly impact ML models. Inconsistent attack labels may confuse algorithms, as some second and third flows are benign while others represent attacks.

- CICFlowMeter does not close a TCP communication aborted by an "RST" flag. So any subsequent communication that has the same identification would be absorbed into the flow that normally would have gotten aborted with the "RST" flag.

## 4.3. Dataset cleansing

Before training the ML models, we performed data preprocessing to address the issues present in the dataset.

### 1) Duplicate Feature Removal

Based on [30] and examining the dataset ourselves, we identified and removed any duplicated features, features such as "Avg Fwd Segment Size" and "Fwd Packet Length Mean" contained identical values. For these cases, we retain only one of them to reduce the dimensionality without losing information. We dropped the features "Packet Length Mean", "Fwd Header Length.1", "Avg Fwd Segment Size", "Avg Bwd Segment Size".

### 2) Removing Error Features

Additionally, we found that there are features that either contains a lot of zero, infinity, negative values when positive ones are expected, or have empty values. This matches with the discovery in [30] where certain features were found to be inaccurately calculated due to errors in the tool used to generate the dataset. As these features may cause problem and introduce noise into the model, we chose to remove them. We removed features that had over 90% of their entries populated with zero or negative values. For features where negative values did not make sense, we dropped those specific rows with negative values, such as "Init_Win_bytes_backward", which represents number of bytes sent in the initial backward window. For empty and infinity values, we only removed the rows that have them.

### 3) Label encoding

The dataset has a combination of numerical and categorical values, so label encoding technique is used so that it is easier and faster for the learning models to process. Afterwards, the label "Benign" is set to 0 and "DDoS" to 1.

At the end of this stage, we have 137329 data objects left, consisting of 55851 instances of "Benign" and 81478 instances of "DDoS".

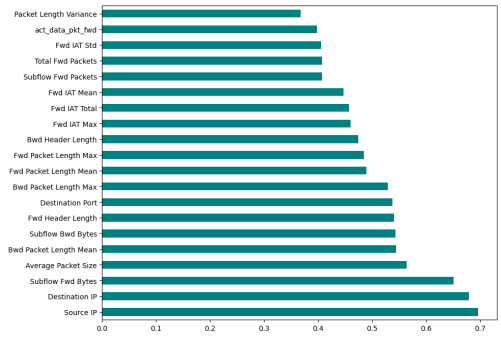


**Fig. 4:** Selected feature information gain graph.

## 4.4. Feature selection

To improve the effectiveness of the ML models, we reduce the dimensionality of the dataset by selecting the most pertinent and informative data attributes, while removing the redundant ones, or those with high correlation of each other. By reducing the features count, we are reducing the computational complexity, which helps to decrease the execution time of algorithms. The utilized techniques and the criteria for selection are described in this section.

### 1) Information gain

Mutual information, also known as information gain [31], is a metric that assesses how much information one variable has regarding another. It is the decrease in entropy of one random variable due to the knowledge of a different one [32]. High information gain features are considered more relevant and could improve the accuracy and efficiency of ML models. From our analysis, we find that each feature possesses varying degrees of information in relation to the target feature. Some features, such as "ECE Flag Count", "Idle Mean", "Active Std", showed very low or even zero information gain. This indicate that these features have little to no importance, as a result, we dropped them. However, some features have a high level of information gain, for example, "Source IP", "Average Packet Size", "Fwd Header Length", which provide substantial information gain value so they are kept. Given that the highest observed information gain was close to 0.7, we set a threshold of 0.35, half of the maximum value, to select most relevant feature. As a result, we have selected the top 20 features for further analysis (see Fig. 4).

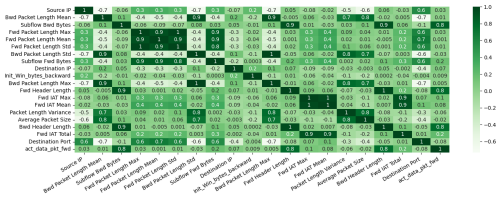

**Fig. 5:** Selected feature information gain graph.

### 2) Correlation

Feature correlation helps us identify both relevant and redundant features by offering insights into their relationships and dependencies. If a feature is relevant only to the target feature and has low correlation with others, it is good [33]. We applied this technique and analyzed the correlations of the 20 selected feature to identify pairs that are highly related. Specifically, we focused on identifying pairs with a correlation value of 1, indicating that one feature is redundant in the presence of the other. Based on Fig. 5, we found some feature pairs that are completely correlated, for example, "Bwd Packet Length Mean" and "Bwd Packet Length Max", "Total Fwd Packets" and "Fwd Header Length", etc. For these cases, we removed one feature from each pair, leaving us with 16 features that exhibited no perfect correlation to each other.

### 3) Data normalization

Data normalization, also known as the Min-Max scaling, is the process of scaling each input variable separately in the range between 1 and 0. We have applied the Min-Max method in Eq. 5 on each of the remaining features to attempt to enhance the efficiency and dependability of the machine learning models, this means that the data were standardized for each feature. This type of technique helps with improving the model performance, reducing the impact of outliers, and ensuring that the data is all on the same scale.

$$X_{scaled} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \times (\max_{new} - \min_{new}) + \min_{new} \quad (5)$$

where $X_{scaled}$ is the scaled value of $x$; $x_{\max}$ and $x_{\min}$ are the minimum and maximum values of $x$ in the original dataset, respectively; $\max_{new}$ and $\min_{new}$ are the minimum and maximum values of the desired range for the scaled data (commonly 0 and 1), respectively.

## 5. AI accuracy improvement and feature analysis

For this work, we aimed to improve the accuracy of ML models for DDoS attack detection, as well as analyzing the features in the dataset to explore their influential factors. In this section, we shall discuss our findings, while using the works done by Saman Sarraf in [8] and Amer A. Abdulrahman in [9] as a comparison.

In [9], the authors used a combination of ML models, those are C5.0, Naive Bayes, SVM and RF, to classify DDoS attack from Benign traffic over the CICIDS2017 dataset, they used 225711 samples along with 80 features for the

 

study. They start by pre-processing the dataset, several data normalization approaches were applied on the numerical features, such as min-max, z-score, and decimal scaling normalization. Afterward, they conducted feature selection to find features that are significant so they could ignore the irrelevant ones from the dataset, this would help reduce dimensionality, enhance processing speed, and save time and resource. Based on the information gain method, their study found 10 most important features to perform the classification, such as "Fwd.IAT.Total", "Flow.IAT.Max", etc. The ML algorithms were then applied on the 10 selected features. The results from the classification show that RF and C5.0 outperformed other algorithms, with average accuracy of 86.80%, 86.45% respectively, and precision score for both was approximately 99%. Meanwhile, SVM had very high false positive rate at 75%. They concluded that the quantity of features and training data samples determines the complexity of classification algorithms. More features would mean that more training data would be needed.

The work in [8] attempted to detect DDoS attack by using Decision Tree and SVM over a significant fraction of the CICIDS2017 dataset, which included 200,000 samples and 84 data features. First, the samples of DDoS and Benign class was randomly divided and shuffled, 100,00 samples from each class were selected to form the dataset for the experiment. The next step involves pre-processing the data. To speed up training and enable faster model convergence, the author decided to apply one-hot encoding to convert categorical values to numeric ones, convert the timestamp feature to absolute total second values, and map the data into an interval with data normalization. Afterwards, the author applied feature correlation analysis to assess the importance of each feature, it was noted that "Flow ID", "SYN Flag Cnt" and "Dst IP" are among the most practical features to identify DDoS in the dataset. Finally, the dataset was divided into two parts: 75% for training and 25% for testing, using the machine learning models to classify the data with every of its features included. The result of the accuracy was very good, it appears that the SVM scored 99.97% accuracy, while Decision Tree scored 100%. However, even though the outcomes of the study were excellent, they were obtained by utilizing every features in the dataset, which may have required more time and money than was necessary.

## 5.1. Features highlight

From what we have gathered after processing the data, the remaining features play the most crucial role in determining if the traffic show sign of a DDoS attack. This means that we would only need to use those features to train our model and ignore the rest of the other ones, this will help reduce the burden on the time and resources needed.

From the information gain table (see Tab. 1), it is shown that the Source IP is the most important feature with roughly 0.68 score. This is most likely due to denial of service can be caused by an address overloading the server by sending multiple requests at short interval. Packet length-related features have an average score of 0.62, this is because DDoS attack can cause long packets, which would consume network resources that could be used by legitimate users. The same can be said for features relating to packet size and number of bytes or packets being sent out.

**Tab. 1:** Parameters of two controllers.

| Features Name | Information Gain |
|---|---|
| Source IP | 0.687346 |
| Bwd Packet Length Mean | 0.67414 |
| Subflow Bwd Bytes | 0.673551 |
| Fwd Packet Length Mean | 0.66937 |
| Fwd Packet Length Std | 0.669333 |
| Bwd Packet Length Std | 0.668721 |
| Subflow Fwd Bytes | 0.668364 |
| Destination IP | 0.660092 |
| Initial Win Bytes Bwd | 0.656791 |
| Fwd IAT Mean | 0.521564 |
| Packet Length Variance | 0.519593 |
| Average Packet Size | 0.515128 |
| Bwd Header Length | 0.514985 |
| Fwd IAT Total | 0.506117 |
| Destination Port | 0.504383 |
| Actual Data Packet Fwd | 0.504362 |

## 5.2. Prediction result

The dataset is partitioned into three subsets to assess the effectiveness of each algorithm. The algorithms are trained using 60% of the dataset, while 20%is allocated for testing purposes and the other 20% is designated for validating. The experiment is implemented using Python and its libraries on Google Collaboratory. Following the splitting of the datset into different subsets, we applied the LR, RF, GNB, MNB, and MLP classification algorithm over the subsets individually. Ultimately, the results for each classification techniques are obtained and analyzed. The scores of the algorithms are derived from the number of correct predictions, they are denoted as True Positives and True Negatives (TP and TN), as well as the number of incorrect predictions, denoted as wrong Positives and False Negatives (FP and FN), and False Positive Rate (FPR) (see Tab. 2).

**Tab. 2:** Parameters of two controllers.

| Algo-rithms | TP | TN | FP | FN | FPR |
|---|---|---|---|---|---|
| LR | 81382 | 55809 | 42 | 96 | 0.000752 |
| RF | 81476 | 55851 | 0 | 2 | 0 |
| GNB | 81375 | 55851 | 0 | 103 | 0 |
| MNB | 81348 | 51066 | 4785 | 130 | 0.085674 |
| MLP | 81475 | 55851 | 0 | 3 | 0 |

The metrics we use for assessing the algorithms are Accuracy, Precision, Recall, F1 score, and the Mean Squared Error (MSE) all of which are calculated based on the formulas in Eq. 6, Eq. 7, Eq. 8, Eq. 9 and Eq. 10.

$$Accuracy = \frac{TP + TN}{TP + TN + FB + FN} \tag{6}$$

$$\text{Re}call = \frac{TP}{TP + FN} \tag{7}$$

$$\text{Pr}\,ecision = \frac{TP}{TP + FB} \tag{8}$$

$$F1 = 2 \times \frac{\text{Pr}\,ecision \times \text{Re}call}{\text{Pr}\,ecision + \text{Re}cal} \tag{9}$$

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} (y_i - p_i)^2 \tag{10}$$

Where $n$ is the number of observations; $y_i$ is the true value of observation $i$; $p_i$ is the predicted value for observation $i$.

Upon examining the results, both RF and MLP models demonstrate a negligible number of occurrences of FP and FN, practically zero. In contrast, the MNB model has the highest occurrences for both FN and FP, with a total of 130 and 4785, respectively. The GNB model exhibits better results, having just 103 instances of FN and zero instances of FP. In the result, FPR is also computed for each model. The results show that all FPRs are smaller than 1% which are accepted in average industrial benchmarks. Next, we assess their score for the metrics mentioned previously in their training, testing, and validation phase.

**Tab. 3:** Parameters of two controllers.

| Model | Train | Test | Validation |
|---|---|---|---|
| LR | 0.9991 | 0.9989 | 0.9986 |
| RF | 1 | 1 | 0.9999 |
| GNB | 0.9992 | 0.9994 | 0.999 |
| MNB | 0.964 | 0.9626 | 0.9662 |
| MLP | 0.9999 | 0.9999 | 0.9997 |

From the result in Tab. 3, it is evident that all of the algorithms achieved either near-perfect or perfect scores for the accuracy metric. RF has the best score overall with 99.99% accuracy during validation and 100% during training and testing. MLP closely follow behind it with 99.97% accuracy over validation, while MNB has the lowest score at around 96.62%accuracy. It is shown that LR and MNB algorithms have strong results in training, reaching 99.91%and 96.4% respectively, however when applied to over the testing set, their scores have a slight decrease by around 0.02% for LR and 0.14% for MNB. It's interesting to note that, in contrast to LR, which drops to 99.86%, the MNB score improves during the validation phase, reaching 96.62%. Moreover, these accuracy rates all are in the range of 90% and 99% which is adopted as an average industrial benchmark.

Meanwhile, the GNB algorithm performs far better than MNB, achieving 99.9% accuracy throughout the validation set. The recall score presented in Tab. 4 demonstrates some slight fluctuations in the scores of some of the algorithms, RF and MNB demonstrate a consistent performance with a slight reduction. For this

 

**Tab. 4:** Parameters of two controllers.

| Model | Train | Test | Validation |
|---|---|---|---|
| LR | 0.9988 | 0.9992 | 0.9984 |
| RF | 1 | 1 | 0.9998 |
| GNB | 0.9987 | 0.9991 | 0.9983 |
| MNB | 0.9985 | 0.9984 | 0.9978 |
| MLP | 0.9999 | 1 | 0.9998 |

metric, RF and MLP continue to have the best score over validation, both of them have 99.98% recall score. LR achieved 99.88% recall score over training set, the score increases to 99.92% over testing, but decreases to 99.84% during validation. With a training score of 99.85%, MNB appears to have the lowest score once again; nevertheless, this score reduces somewhat to 99.84% during testing, and then drops even further to 99.78% during the validation set. The GNB algorithm performs somewhat better compare to its MNB variant; it starts off at 99.87% during the training set and rises to 99.91% in testing; however, during validation, the score drops to 99.83%.

The precision score from Tab. 5 shows that the RF and GNB outperformed the other algorithms, scoring 100% on all three subsets. Both LR and MNB have training scores of 99.97% and 94.4%, respectively, they exhibit a decrease during testing, with LR decrease slightly to 99.89% and MNB at 94.19%, however, their score improve slightly during validation, with LR rising to 99.92% and MNB at 94.8%. On the other hand, MLP achieves consistent score of 99.99% during training and testing, it only has a minor decrease to 99.97% in the validation phase.

**Tab. 5:** Parameters of two controllers.

| Model | Train | Test | Validation |
|---|---|---|---|
| LR | 0.9997 | 0.9989 | 0.9992 |
| RF | 1 | 1 | 1 |
| GNB | 1 | 1 | 1 |
| MNB | 0.944 | 0.9419 | 0.948 |
| MLP | 0.9999 | 0.9999 | 0.9997 |

The F1 score results in Tab. 6 show that RF has the best score once again, it achieves 100% over training and testing, with a minor decrease to 99.99% over validation. In this metric, MLP is the second highest by persistently achieving 99.99% score during training and testing, with a slight decrease to 99.98% over validation. The MNB model has a score of 97.05%over the train set, it decreases to 96.94% during testing, but rises to 97.23% over the validation set. Meanwhile, GNB has a score of 99.93% during training, it has a tiny increase during testing at 99.95%, but decreases to 99.91% in validation. LR consistently decreases over the three subsets; it begins at 99.92% during training, drops to 99.9% during testing, and finally drops to 99.88% during validation.

**Tab. 6:** Parameters of two controllers.

| Model | Train | Test | Validation |
|---|---|---|---|
| LR | 0.9992 | 0.999 | 0.9988 |
| RF | 1 | 1 | 0.9999 |
| GNB | 0.9993 | 0.9995 | 0.9991 |
| MNB | 0.9705 | 0.9694 | 0.9723 |
| MLP | 0.9999 | 0.9999 | 0.9998 |

In Tab. 7, the mean squared error of the algorithms is shown, we can see that each model has very low error measured. LR has consistently low MSE values across train, test, and validation sets, with values around 0.0018. Meanwhile, RF demonstrated no error, with an MSE of 0 across all subsets. Similarly, GNB has very low error, with an MSE of approximately 0.00002 across all sets. The highest MSE values is seen with the MNB model, at 0.227, indicating a greater prediction error. Finally, the MLP showed extremely low error, with MSE values close to 0.

**Tab. 7:** Parameters of two controllers.

| Model | Train | Test | Validation |
|---|---|---|---|
| LR | 0.0018 | 0.0019 | 0.0017 |
| RF | 0 | 0 | 0 |
| GNB | 0.00002 | 0.00002 | 0.00002 |
| MNB | 0.227 | 0.2257 | 0.2278 |
| MLP | 0.000014 | 0.000022 | 0.00002 |

Overall, the ML models employed in this study demonstrated excellent performance across all evaluated metrics, indicating high accuracy and minimal error.

### 5.3. Result comparison

Because of the danger that DDoS poses to information security, many research works have been done in applying ML techniques for intrusion detection using various datasets, some of those works shared the same dataset and algorithms used for this work. In this section, we shall make comparison, where possible, between the results that we obtained and the ones found in the previous works.

The comparison between the prediction results of the shared algorithms and assessment metric of our work and those of [9] can be found in Tab. 8. Upon inspection, it is evident that we have made some improvements in the accuracy, precision, and recall metrics for the shared algorithms. We are unable to make comparison for the F1 score metric because it was not considered in their work, and we included both GNB and MNB since we are unaware of the type of Naive Bayes that was employed in their research.

**Tab. 8:** Parameters of two controllers.

| | Algorithm | Accuracy | Precision | Recall |
|---|---|---|---|---|
| Present | RF | 0.9999 | 1 | 0.9998 |
| Present | GNB | 0.999 | 1 | 0.9983 |
| Present | MNB | 0.9662 | 0.948 | 0.9978 |
| [9] | RF | 0.868 | 0.9963 | 0.8629 |
| [9] | Naive Bayes | 0.79996 | 0.86031 | 0.90069 |

Looking over both of the results of RF, it can be seen that we have managed to raise the accuracy score from 86.8% to 99.99%, the recall score has grown to 99.98% from 86.29%, and the precision score has reached 100% from 99.63%. For the Naive Bayes model, both of our GNB and MNB results have achieved better scores, with accuracy having 79.99% vs our 99.9% and 96.62%, the recall has 90.06% vs our 99.83% and 99.78%, and precision has 86.03%vs our 94.8% and 99.97%. It is possible that because we considered the limitations of the dataset, we were able to obtain a better classification result than they were.

For the study conducted in [8], we compare the most significant features that were identified in their study and ours. The selected essential features in their study were "Flow ID", "SYN Flag Count" and "Destination IP". Meanwhile, our study identifies "Source IP", "Bwd Packet Length Mean" and "Subflow Bwd Bytes" as the top three features, however "Destination IP" is also in our list of key features. The author of the study failed to account for the dataset limitations identified by Arnaud Rosay and colleagues in their previous work [19]. This oversight, along with the difference in dataset version used may have contributed to discrepancies between our findings and theirs.

In the work done by Maria Rodriguez and her team in [10], although the study is for classifying the entire CICIDS2017 dataset and not just the DDoS portion, their result is still interesting so we will attempt to compare between ours results and their binary classification with feature selection results, because that is the most similar one out of the three strategies that they employed. Since only the F1 results were shown in their paper, we shall compare our F1 score with theirs. The results comparison can be found in Tab. 9.

**Tab. 9:** Parameters of two controllers.

| | Algorithms | F1 |
|---|---|---|
| Present | GNB | 0.9991 |
| Present | MNB | 0.9723 |
| Present | MLP | 0.9998 |
| Present | LR | 0.9988 |
| Present | RF | 0.9999 |
| [10] | Naive Bayes | 0.504 |
| [10] | MLP | 0.605 |
| [10] | LR | 0.577 |
| [10] | RF | 0.976 |

Although the goal of our work and [10] is different, the result still have some similar trend that parallel with each other. The tree-based RF algorithm has the highest score for both side, with ours has 99.99% and 97.6% in their study. Following along, MLP has the second highest result with 99.98% and 60.5%. Next, our LR scored 99.88%, and theirs scored 57.7%. As for the last algorithms, they scored 50.4% for the Naive Bayes results, compared to 99.91% and 97.23% for our GNB and MNB, respectively. The reason we have such different result is because our models are only applied to the DDoS

portion of the CICIDS2017 dataset, whereas the models in their paper are applied to every attack file in the dataset. Furthermore, the excellent performance result of RF on both sides suggests that it might be the ideal algorithm for this kind of task.

### 5.4. Limitations

Despite the near perfect performance of the ML models employed in this study, there are still limitations and challenges warrant consideration. This study did not address real-time detection, hence, this need to be looked at in future work. The scalibility factor presents another possible issue, the models performed well on the utilized dataset, but they may not perform as well on a more complex dataset. Additionally, the dataset employed is outdated and contains numerous deficiencies, which may impact the effectiveness of the models in practical environments. Furthermore, this study was conducted using only the CICIDS2017 dataset, which lead to the generalization and reliability of the models to be uncertain. Therefore, we recognized that testing across multiple datasets would further validate the robustness and generalizability of the employed models.

## 6. Conclusion and future works

This paper is an attempt to improve the accuracy of DDoS detection with ML algorithm, while using the studies in [8], [9], and [10] as a base for comparison. In terms of the accuracy, precision, recall, and F1 score of the ML algorithms that we employed, which are LR, RF, Naive Bayes (including Multinomial and Gaussian), and MLP, we have successfully attained remarkably high scores, ranging from 94.8%to a flawless 100%. This is a noticeable increase compare to the results collected in the two studies [8] and [9]. Our findings also indicate that features, such as source IP, packet length, size-related attributes, and features relating to packet count, play a crucial role in the identification of DDoS attacks.

For future investigations of this subject, there exist opportunities for enhancement and expansion of the present study, which have a number of limitations. One of which is the deployment of the models in a real-time network environments to evaluate their effectiveness in detecting attacks under dynamic conditions. When we deploy it to the real time environment, the dynamic feature of realtime data collection, realtime data preprocessing, and the volumne size of the collected live data can be a challenge.

Additionally, since the results of the study is only from a single dataset, we can use additional datasets from different sources to further validate the generalizability and robustness of the employed models. Furthermore, the future work could utilize a more up-to-date and varied network traffic dataset to address issues of the one used in this study. Lastly, we could also perform more intricate data analysis, and employ alternative algorithms and data mining methodologies, among other potential avenues for improvement.

## References

[1] A. Petrosyan. Number of internet and social media users worldwide as of april 2023. *https://www.statista.com/*, 2023.

[2] W. Stallings. Cryptography and network security: Principles and practice, 5th edition. *Pearson*, pages 9–13, 2010.

[3] S. Chakraborty, P. Kumar, and B. Sinha. A study on DDoS attacks, danger and its prevention. *IJRAR*, 6(2):10–15, 2019.

[4] K. Arora, K. Kumar, and M. Sachdeva. Impact analysis of recent DDoS attacks. *Comput. Sci. Eng.*, 3, 2011.

[5] K. Garg and R. Chawla. Detection of DDoS attacks using data mining. *Int. J. Comput. Bus. Res.*, 2(1), 2011.

[6] K. Borisenko, A. Rukavitsyn, A. Gurtov, and A. Shorov. Detecting the origin of DDoS attacks in openstack cloud platform using data mining techniques. *Int. Conf. on Next Gener. Wired/Wireless Netw. Conf.*

*on Internet Things Smart Spaces*, pages 303–3015, 2016.

[7] R. Doshi, N. Apthorpe, and N. Feamster. Machine learning ddos detection for consumer internet of things devices. *IEEE Secur. Priv. Work. (SPW), San Francisco, CA, USA*, pages 29–35, 2018.

[8] S. Sarraf. Analysis and detection of DDoS attacks using machine learning techniques. *Am. Sci. Res. J. for Eng. Technol. Sci.*, 66:95–104, 2020.

[9] A.A. Abdulrahman and M.K. Ibrahem. Evaluation of ddos attacks detection in a CICIDS2017 dataset based on classification algorithms. *Iraqi J. Inf. Commun. Technol.*, 1:49–55, 2018.

[10] M. Rodriguez, A. Alesanco, L. Mehavilla, and J. Garcia. Evaluation of machine learning techniques for traffic flow-based intrusion detection. *Sensors*, 22, 2022.

[11] University of California. KDD cup 1999 dataset. *Univ. Calif.*, 1999.

[12] S. Wafa, I. Ibrahim, , and S. Sumia. Detection of communication network intruders using artificial neural networks. *3rd Int. Maghreb Meet. Conf. on Sci. Tech. Autom. control Comput. Eng. Benghazi, Libya*, 2023.

[13] S. Mishra. Blockchain and machine learning-based hybrid IDS to protect smart networks and preserve privacy. *Electronics*, 12(16):3524, 2023.

[14] M. Tavallaee, E. Bagheri, W. Lu, and A.A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. *IEEE Symp. on Comput. Intell. for Secur. Déf. Appl. Ottawa, ON, Can.*, page 1–6, 2009.

[15] V.B. Reddy, K.S. Basha, D. Roja, N.R. Purimetla, S.S. Vellela, and K.K. Kumar. Detection of ddos attack in iot networks using sample elected rnn-elm. *Int. Conf. on Recent Adv. Sci. Eng. Technol. (ICRASET)*, 2023.

[16] CAIDA Datasets. Overview of datasets, monitors, and reports. *Cent. for Appl. Internet Data Anal.*, 2009.

[17] A. Ferriyan, A.H. Thamrin, K. Takeda, and J. Murai. Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Appl. Sci.*, 11, 2021.

[18] Canadian Institute for Cybersecurity. Intrusion detection evaluation dataset 2017. *Can. Inst. for Cybersecur.*, 2017.

[19] I. Sharafaldin, A.H. Lashkari, and A.A. Ghorbani. A detailed analysis of the CICIDS2017 data set. *Int. Conf. on Inf. Syst. Secur. Privacy, 2018, Commun. Comput. Inf. Sci.*, 997:172–188, 2019.

[20] M. Alkasassbeh, A. Hassanat, G.A. Naymat, and M. Almseidin. Detecting distributed denial of service attacks using data mining techniques. *Int. J. Adv. Comput. Sci. Appl.*, 7(1), 2016.

[21] K. Kumari and M. Mrunalini. Detecting denial of service attacks using machine learning algorithms. *J. Big Data*, 9:56, 2022.

[22] M. Azmi, C. Foozy, K. Sukri, N. Abdullah, I.A. Hamid, and H. Amnur. Feature selection approach to detect DDoS attack using machine learning algorithms. *Int. J. on Informatics Vis.*, 5:395, 2021.

[23] P. Kamboj, M.C. Trivedi, V.K. Yadav, and V.K. Singh. Detection techniques of DDoS attacks: A survey. *2017 4th IEEE Uttar Pradesh Sect. Int. Conf. on Electr. Comput. Electron. (UPCON), Mathura, India*, pages 675–679, 2017.

[24] M.D. Prasad, P.V Babu, and C. Amarnath. Machine learning DDoS detection using stochastic gradient boosting. *Int. J. Comput. Sci. Eng.*, 7:157–166, 2019.

[25] A. Subasi. Practical machine learning for data analysis using python. *Acad. Press*, 2020.

[26] J.V. Plas. Python data science handbook: Essential tools for working with data. *O'Reilly Media*, page 382–390, 2017.

 

[27] C. Sammut and G.I. Webb. Encyclopedia of machine learning. *Springer*, pages 739–740, 2010.

[28] S. Trenn. Multilayer perceptrons: Approximation order and necessary number of hidden units. *IEEE Trans. On Neural Networks*, 19, 2008.

[29] Canadian Institute for Cybersecurity. CICFlowmeter (formerly ISCXFlowmeter). *https://www.unb.ca/cic/research/applications*, 2017.

[30] A. Rosay, E. Cheval, F. Carlier, and P. Leroux. Network intrusion detection: a comprehensive analysis of CIC-IDS2017. *Proc. 8th Int. Conf. on Inf. Syst. Secur. Priv. (ICISSP 2022)*, page 25–36, 2022.

[31] Y. Wu and A. Zhang. Feature selection for classifying high dimensional numerical data. *Proc. 2004 IEEE Comput. Soc. Conf. on Comput. Vis. Pattern Recognition, 2004*, 2, 2004.

[32] T.M. Cover and J.A. Thomas. Elements of information theory. *Wiley Inter science*, page 18, 2006.

[33] L.Yu and H. Liu. Feature selection for high dimensional data: A fast correlation-based filter solution. *Proc. Twentieth Int. Conf. on Mach. Learn. (ICML-2003), Wash. DC*, 2:856–863, 2003.

 

## About Authors

**Le Ba NGUYEN** is a junior researcher currently for SEcurity, Privacy and Big DatA (SEA) Lab in Vietnamese - German University (VGU). He obtains a B.S. Degree in Computer Science at VGU in April 2024. His research interests include data science, machine learning and their wide range of applications.

**Quoc-Binh NGUYEN** is currently working as a lecturer in Faculty of Information Technology, Ton Duc Thang University. He obtained the Bachelor Degree in Information Technology in 2009 and Master Degree in Computer Science in 2012 at University of Science, VNU of Ho Chi Minh City.

**Ngoc Hong TRAN** has been a lecturer at the Vietnamese German University (VGU), Vietnam, since 2016 and head of SEcurity, Privacy and Big DatA (SEA) Lab, VGU. She obtained a Bachelor Degree in Information Technology in 2004, and a Master Degree in Computer Science in 2008, from University of Science, Vietnam National University - Ho Chi Minh City. She achieved a PhD Diploma in Computer Science from University of Insubria, Italy in 2016. In addition, she had been a lecturer at the University of Science, VNU-HCM (2004-2016). She was an exchange scholar in Portland State University (PSU), Portland City, Oregon State, USA, in 2010. She was an invited researcher in National Institute of Informatics, Tokyo, Japan (2009), in LAAS-CNRS, Toulouse, France (2012), in Singapore University of Technology and Design (SUTD), Singapore, March-April 2017, and was a postdoctoral researcher in University College Dublin, Ireland (2018-2021). She has been a reviewer of several international conferences. Her research interests are data science, machine learning, security and privacy in a variety of scenarios in smart healthcare, smart manufacturer, education digitalization, block chain, IoT, mobile social network, and web composition.

 