Anomaly Detection in Networks Using Machine Learning

Thesis · August 2018

CITATIONS
11

READS
8,277

1 author:

Kahraman Kostas
Heriot-Watt University
5 PUBLICATIONS **11** CITATIONS

SEE PROFILE

Anomaly Detection in Networks Using Machine Learning

Kahraman Kostas

A thesis submitted for the degree of Master of Science in Computer Networks and Security

Supervisor: Dr. Martin Reed
School of Computer Science and Electronic Engineering
University of Essex

August 2018

Abstract. Every day millions of people and hundreds of thousands of institutions communicate with each other over the Internet. In the past two decades, while the number of people using the Internet has increased very fast. Parallel to these developments, the number of attacks made on the Internet is increasing day by day. Although signature-based methods are used to prevent these attacks, they are abortive against zero-day attacks. On the other hand, the Anomaly-based approach is an alternative solution to the network attacks and has the ability to detect zero-day attacks as well. In this study, it is aimed to detect network anomaly using machine learning methods. In this context, the CICIDS2017 has been used as dataset because of its up-to-datedness, and wide attack diversity. On this dataset, feature selection was made by using the Random Forest Regressor algorithm. Seven different machine learning algorithms have been used in the application step and achieved high performance. Machine learning algorithms and success rates are as follows: Naive Bayes 86%, QDA 86%, Random Forest 94%, ID3 95%, AdaBoost 94%, MLP 83%, and K Nearest Neighbours 97%

Keywords: Machine Learning, Network Security, Network Anomaly Detection, CICIDS2017, Naive Bayes, QDA, Random Forest, ID3, AdaBoost, MLP, KNN

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Every day millions of people and hundreds of thousands of institutions communicate with each other over the Internet. In the past two decades, while the number of people using the Internet has increased very fast, today this number has exceeded 4 billion and this increase is continuing rapidly[1, 2].

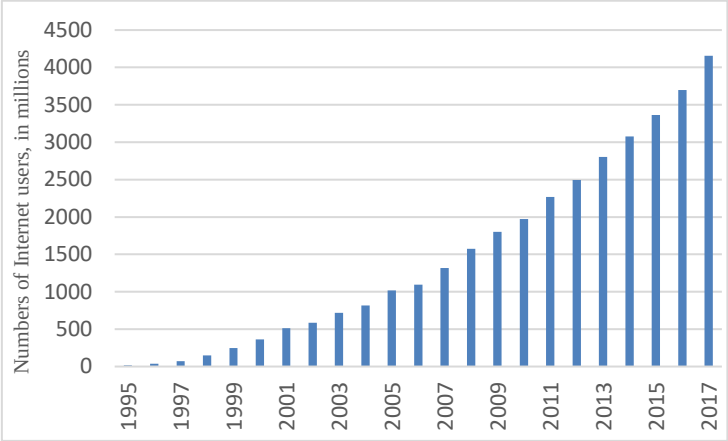


Figure 1. Increase in The Number of the Internet Users According to Years (1995-2017) [1, 2].

Parallel to these developments, the number of attacks made on the Internet is increasing day by day. Against these attacks, there are two basic methods used to detect the attacks in order to ensure information security; identification based on signature, and detection based on anomaly.

Signature-based methods use the database they created to detect attacks. This method is quite successful, but the databases need to be constantly updated and new attack information processed. Moreover, even if the databases are up-to-date, they are vulnerable to the zero-day (previously unseen) attacks. Since these attacks are not in the database, they cannot prevent these attacks. The anomaly-based approach focuses on detecting unusual network behaviours by examining network flow. This method, which has been successful in detecting attacks that it has not encountered before, so is effective against zero-day attacks[1, 3].

In addition, more than half of today's internet usage is encrypted using SSL / TLS (Secure Sockets Layer / Transport Layer Security) protocols, and this rate is increasing day by day[4]. Because of the inability to observe the contents of the encrypted internet stream, signature-based methods do not work effectively on this type of data. However, the anomaly-based approach analyses data by its general properties such as size, connection time, and number of packets. So, it does not need to see the message content and it can also do the analysis of encrypted protocols. Due to all these advantages, the anomaly-base detection method is being used intensively to detect and prevent network attacks.

In this study, it was aimed to contribute to the literature by developing a system that detects network anomaly quickly and effectively by means of machine learning methods.

1.2 Goals and Objectives

1.2.1 Goals

The goals that are aimed to achieve at the end of this study are as follows [1]:

- Examination of machine learning algorithms that can be used to detect network anomalies.
- To detect network attacks in a fast and effective way by studying network anomaly with machine learning methods.
- To determine the success level of the study by comparing the results obtained in it with the studies previously conducted in this area.
- Contributing to the literature by obtaining close results from previous studies in the detection of network anomalies.

1.2.2 Objectives

The objectives that are aimed to achieve at the end of this study are as follows [1]:

- To examine the previous work done in the field by doing extensive field research.
- Selecting the appropriate dataset by performing comprehensive research on the alternatives to the dataset.
- Choosing suitable algorithms by conducting extensive research on machine learning algorithms.
- Deciding on right algorithms by performing exhaustive research on machine learning methods.
- Selecting the appropriate software platform.
- Choosing the suitable hardware/equipment platform.
- Deciding on the right evaluation criteria.
- Choosing the benchmark studies to be compared during the evaluation phase.

1.3 Structure of The Report

In the Background and Related Work section, some preliminary information is provided for a better understanding of the work such as data sets, anomaly and attack types, detailed information about attacks and machine learning algorithms. Finally, similar studies in the literature are listed and given information about them.

The Methodology section consists of two subsections. The first sub-section, Tools and Methods, provides information about the software and hardware used during the work and describes the performance evaluation methods. The second step, the implementation, contains these sub steps; data cleaning, the division of the data into training and testing, implementation of feature selection and machine learning algorithms.

In the Results and Discussion section, the results obtained in the implementation step are shared and their cause-effect relationships and alternative approaches are discussed.

In the Evaluation section, the data obtained in the study is compared with another study selected from the literature.

In the Conclusion and Future Work section, the study is briefly summarized and possible developments and innovation which could be applied in the study are touched.

2 Background and Related Work

2.1 Datasets

In the detection of network anomaly by machine learning methods, there is a need for a large amount of harmful and harmless network traffic for training and testing steps. However, it is not possible for real network traffic to be used publicly because of privacy issues. To meet this need, many datasets have been produced and continue to be produced. In this step, information will be given about some popular datasets, after that, they will be compared and evaluated to decide which of them to use in the implementation phase.

2.1.1 DARPA 98 [5]

With this dataset created by MIT Lincoln laboratory with DARPA funding, it is aimed to create a training and testing environment for Intrusion Detection Systems. In this dataset, the United States Air Force's local computer network is simulated. The data stream consists of processes such as file transfer via FTP, internet browsing, sending and receiving e-mail and IRC messages. In addition to Benign/Normal network traffic, it includes 38 attacks that can be grouped under attack types such as Denial of Service (DoS), User to Remote (U2R), Probe, and Remote to Local (R2L)_[6].

This dataset has received a lot of criticism, especially not including the fact that it does not reflect real-world network traffic, it is no longer up-to-date and not include flows that can be classified as false positives (the benign data classified as attack, false alarm)[7]. However, the DARPA98 dataset is still important because it was used as a source for the creation of commonly used datasets such as KDD Cup 99 and NSL-KDD.

2.1.2 KDD 99[8]

This dataset was created by the University of California, Irvine for use by intrusion detection systems in The Third International Knowledge Discovery and Data Mining Tools Competition (The KDD Cup '99). The data packets that make up the DARPA98 dataset are used. 21 properties have been created by applying feature extraction process to be used by machine learning methods[4].

It is divided into two parts as training part and test part. The training section consists of 4898431 and the test section consists of 311029 data streams. KDD99 contains 38 attack types. Of these attacks, 14 are only specific to the test section and represent unknown attacks. Thus, the detection of unknown attacks on the test section can also be controlled.

Compared to the DARPA99 dataset, KDD99, which is more suitable for machine learning methods with both the new feature system and training and data parts, has been preferred in many studies [9].

2.1.3 **CAIDA [10]**

CAIDA (Centre of Applied Internet Data Analysis) is an organization engaged in internet data analysis. The dataset provided by the facilities of this organization is referred to by the same name. The dataset that makes up this dataset comes from a few hours of data flow recording of the OC48 backbone connection over San Jose city. This dataset also contains a section that simulates an hourly DDoS attack [7].

In the CAIDA dataset, data flows are exemplified only by specific applications and specific attacks. Therefore, the variety of sampling is quite limited. In addition, in this data set, data streams are not labelled. The fact that the data are unlabelled makes it very difficult to use this dataset in machine learning applications [11].

2.1.4 **NSL-KDD [12]**

Although the KDD99 dataset created in 1999 was a very good alternative to DARPA98, it was observed that there are too many repetitions in KDD99, and these repetitions affect the results of the studies performed with it and the performance of the machine learning algorithms. In addition, because the size of KDD99 is too large, researchers have tried to use part of this data set. Unfortunately, in order to minimize the dataset, randomly selected data from within could not capture all the properties of the data set [9].

To overcome these shortcomings, in 2009, Tavallaee et al. [13] created a new data set called NSL-KDD. In this version, they eliminated the mistakes and repetitions in KDD99. The NSL-KDD dataset consists of 4 parts under two main headings as training and testing: training data (KDDTrain+), 20% of the training data (KDDTrain+_20Percent), test data (KDDTest+) and a smaller version of the test data with all difficulty levels(KDDTest-21)[12].

2.1.5 **ISCX 2012 [14]**

Although many datasets have been used in the detection of anomalies, objections have arisen as these data have lost their up-to-datedness or the synthetic data used do not reflect real-world data. To solve these problems, in 2012, ISCX 2012 (Intrusion detection evaluation dataset) was created using the seven-day Internet stream on the testbed created by the Canadian Institute for Cybersecurity. The highlights of this dataset are[15]:

- Because it was built using real devices, real normal and malicious streams including FTP, HTTP, IMAP, POP3, SMTP and SSH protocols[1] were created.
- All data are labelled.
- The attack variety is high and includes different types of attack (Infiltrating, Denial of Service, Distributed Denial of Service and Brute Force SSH).

[1]

FTP	:	File Transfer Protocol	SMTP	:	Simple Mail Transfer Protocol
SSH	:	Secure Shell	IMAP	:	Internet Message Access Protocol
POP3	:	Post Office Protocol 3	HTTP	:	Hypertext Transfer Protocol

On the other hand, the ISCX 2012 data set does not include SSL / TLS (Traffic Sockets Layer / Transport Layer Security) traffic, which accounts for more than half of today's Internet traffic so giving the impression that it will be inadequate to meet today's needs.

2.1.6 **CICIDS 2017 [16]**

Finally, another dataset to be addressed is the CICIDS 2017 (Intrusion Detection Evaluation Dataset) created by the Canadian Institute for Cybersecurity at the University of New Brunswick. This dataset consists of a 5-day (3rd July- 7th July 2017) data stream on a network created by computers using up-to-date operating systems such as Windows Vista / 7 / 8.1 / 10, Mac, Ubuntu 12/16 and Kali. Details of the dataset can be seen from Table 1[4, 7].

Flow Recording Day (Working Hours)	**pcap File size**	**Duration**	**CSV File Size**	**Attack Name**	**Flow Count**
Monday	10 GB	All Day	257 MB	No Attack	529918
Tuesday	10 GB	All Day	166 MB	FTP-Patator, SSH-Patator	445909
Wednesday	12 GB	All Day	272 MB	DoS Hulk, DoS GoldenEye, DoS slowloris, DoS Slowhttptest, Heartbleed	692703
Thursday	7.7GB	Morning	87.7 MB	Web Attacks (Brute Force, XSS, Sql Injection)	170366
		Afternoon	103 MB	Infiltration	288602
Friday	8.2GB	Morning	71.8 MB	Bot	192033
		Afternoon	92.7 MB	DDoS	225745
		Afternoon	97.1 MB	PortScan	286467

Table 1. Details of CICIDS2017 Dataset

The CICIDS 2017 data set has the following advantages over the other datasets mentioned above[4, 7]:

- The obtained data is the real-world data; was obtained from a testbed consisting of real computers.
- Data streams are collected from computers with the up-to-date operating system. There is operating system diversity (Mac, Windows, and Linux) between both attacker and victim computers.
- Data sets are labelled. In order to apply the machine learning methods, the feature extraction, which is a critical step, was applied and 85 features (see Appendix A for the feature list) were obtained.
- Both raw data (pcap files - captured network packets files) and processed data (CSV files-comma-separated data files) are available to work on.
- In the course of deciding which attack to take place, the 2016 McAfee security report was used, so there is a wide and up-to-date assortment of attacks.
- It is more abundant than other data sets in terms of protocols used. It also includes the HTTPS (Hypertext Transfer Protocol Secure) protocol in addition to FTP, HTTP, SSH and e-mail protocols.

On the other hand, some disadvantages of CICIDS2017 are the following:

- Raw data files and processed data files are very large (47.9 GB and 1147.3 MB respectively).
- Unlike the KDD99 and NSL-KDD datasets, CICIDS2017 does not have separate files dedicated to training and testing. These sections should be created by users. How to do this is handled in the Creation of Training and Test Data section.
- The formation of DARPA98, KDD99 and then NSL-KDD is analogous to an evolutionary process. At each step, faults and deficiencies in the previous dataset were observed and measures were taken to prevent them. On the other hand, CICIDS 2017 is very new and has not been studied much yet, so it is likely to contain some minor mistakes. What these mistakes are and how to fix them is processed in the Data Cleansing section.

In this section, it is decided to select CICIDS2017- processed data (CSV files) as the data set to be used in the implementation phase as a result of the comparison process. The most important factors in this preference are the fact that the dataset is up-to-date and offers a wider protocol and attack pool. In addition, since the number of works done with this data set is still a few, working with this dataset could be considered to be a possible significant contribution to the literature.

2.2 Anomaly and Attack types

2.2.1 Anomaly Types

The anomaly is a sample that does not have well-defined properties of a normal sample[11]. In order for the anomaly to be understood, the rules that make up the normal concept must be specific and valid. The anomaly is analysed under 3 headings:

Point anomaly: If a particular data sample looks different from the whole dataset with the properties it carries, it is called a point anomaly [11]. For example, it is a point anomaly that a person who makes a low amount of shopping every day with a credit card makes a very high amount of shopping on a random day.

Contextual anomaly: The out-of-pattern behaviour of a data sample depends on certain conditions or occurs under certain conditions [11]. For example, it is a contextual anomaly that a person who makes a low amount of shopping every day with a credit card makes a very high amount of shopping on the feast day.

Collective anomaly: If a data heap consisting of similar data has abnormal properties according to normal data, this is called the collective anomaly [11]. For example, it is a collective anomaly that the increase in credit cards spending on the valentine's day.

2.2.2 Network Attacks Types

Network security tries to protect the network from attacks against these three principles: confidentiality, integrity, and availability [11, 17].

Confidentiality: Information should be accessible only to legitimate users and unauthorized access should be prevented.

Integrity: adding, modifying and deleting information can only be done by the legitimate user. Unauthorized persons should not be able to modify information.

Accessibility: The system should always be accessible to the legitimate user.

Network attacks are attempts to violate these 3 essential features. The attacks can be summarized in 4 headings.

Denial of Service (DoS): In this type of attack, an attacker abuses system resources, preventing the legitimate user from taking advantage of the service. The simplest example is to make it drop out of service by sending a large number of requests to a web server [11]. Dos attacks can be divided into two subparts as bandwidth depletion and resource depletion. While bandwidth depletion aims to consume a victim's bandwidth by providing a very high data flow, resource depletion attacks usually try to consume resources such as the victim's memory and processor with a lot of packages [18].

Probe (Information Gathering): These attacks aim at collecting information about the target. Through this attack, attackers can get a lot of important information such as network structure, the operating system used, types and properties of networked devices. Although this attack does not directly affect the system, it is very important because it is preparing the ground for many attacks that can harm the system [11].

U2R (User to Root): In this type of attack, an attacker tries to gain control of the administration account in order to access and steal important resources. the attacker may use system vulnerabilities or brute-force attacks to gain the Administrator account [11].

R2U / R2L (Remote to User / Remote to Local): In this attack, the attacker infiltrates the victim's network to gain a privilege to send packets from the victim computer. An attacker can use system vulnerabilities or brute-force attacks to gain this privilege [11].

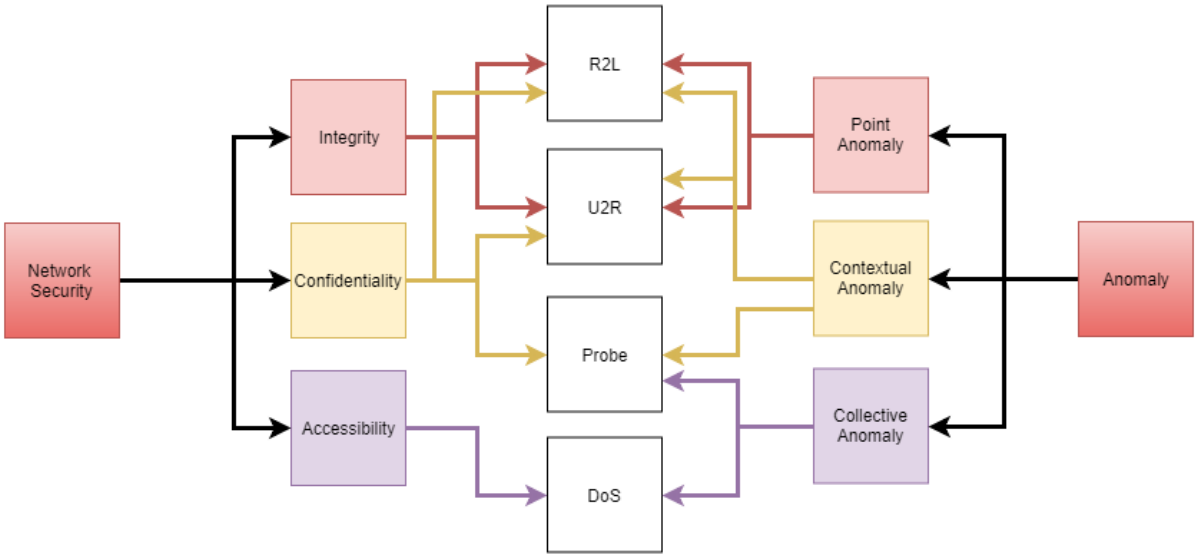


Figure 2. The relationship between network anomalies and network attacks.

Classifying the network attacks according to the anomalies they create, can be useful in detecting the attack. Each attack causes differentiation (an anomaly) in the network flow. For example, DoS attacks are known to increase the amount of data flowing and the number of packets in the network. During a DoS attack, a large number of abnormal streams can be mentioned. Thus, classify DoS attacks as collective anomalies. On the other hand, it would be more appropriate to classify U2R and R2U as contextual and point anomaly because the attack is for a particular user, certain port, and specific purpose. Additionally, the probe attack creates a dense packet traffic on the network, as well as it has a specific purpose. In this context, the probe attack can be defined as a contextual and collective anomaly. The relationship between network anomalies and network attacks is summarized in Figure 2.

2.3 Attacks

In this section, the types of attacks that the data set contains are examined in detail.

All data in the dataset are tagged with 15 labels. One (Benign) of these tags represents normal network movements while the other 14 represent attacks. The benign record, formed using Mail services, SSH, FTP, HTTP, and HTTPS protocols represent a non-harmful / normal data stream on the network, created by simulating real user data [6]. The names and numbers of these labels can be seen in Figure 3.

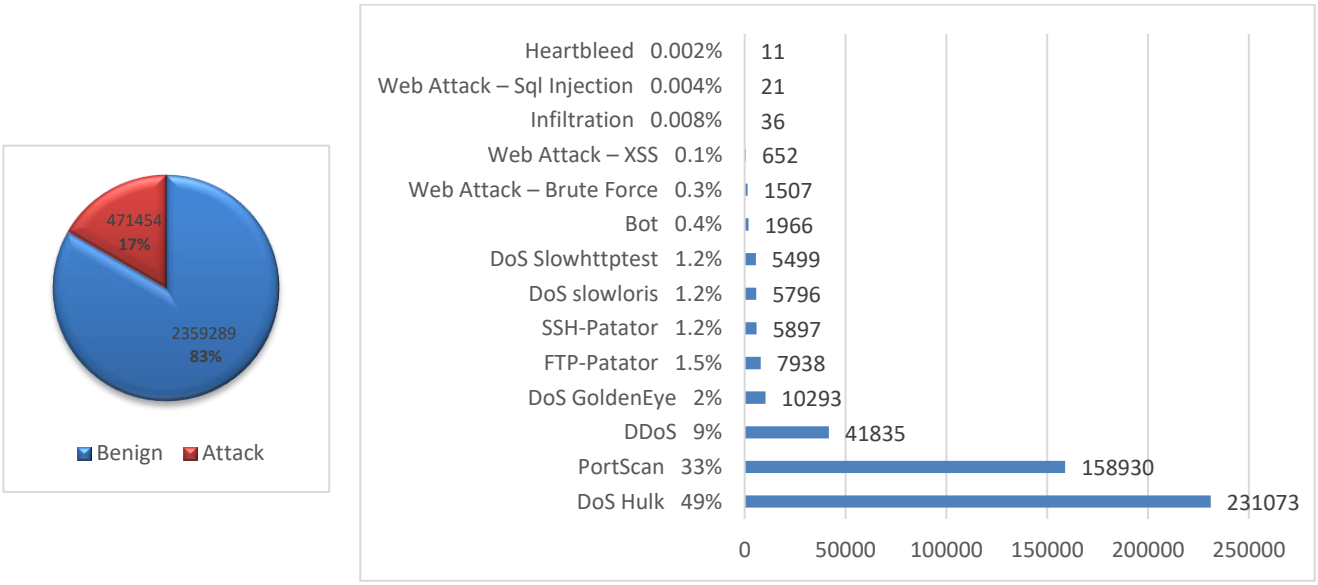


Figure 3. The distribution of data flow and attack types in the dataset.

When the numbers of these attacks are examined, it is immediately apparent that the numbers of some attacks are very high. For example, the DoS HULK attack is almost half of all attacks and the PortScan attack is one-third of all attacks. The reason for this imbalance in the distribution is the nature of these attacks. Both DoS and PortScan attacks cause too much data and packet flows during the attack. Therefore, during these attacks, it is completely natural to observe more intense traffic from normal usage and other types of attacks.

The attacks in the dataset can be explained as follows:

2.3.1 DoS HULK

HULK[19] (HTTP (Hypertext Transfer Protocol) Unbearable Load King) takes place on Wednesday-records with other DoS attacks within the CICIDS2017 data set. HULK can be used to perform both DoS and DDoS attacks. Because this attack influences the server load first-hand, it can disable the server in a very short time like a minute or so. HULK has the capability to conceal the real user agent and use a different template for each attack request. During the attack, it creates TCP-SYN (Transmission Control Protocol - Synchronization) floods and multiple HTTP-GET flood requests. To better understand this attack, TCP-SYN flood and HTTP-GET flood methods can be examined[20].

TCP-SYN Flood

The TCP-SYN flood [21] attack is intended to exploit the 3-way handshake method to consume server resources and render the server out of service. The 3-way handshake method involves the following steps:

- The client that wants to connect to the server sends a SYN packet to the server to start the communication.
- The server receiving the SYN packet sends the corresponding SYN-ACK packet.
- Finally, the client initiates communication by sending an ACK packet to the server.

In the TCP-SYN flood attack, the first two steps occur; the attacker (client) sends the SYN packet to the victim (server), and the server responds with the SYN-ACK packet in response. However, the attacker does not send the ACK packet that should be sent, and the connection waits as incomplete. The server reserves a resource by keeping these requests in a log queue. When the number of requests increases, the server becomes inaccessible.

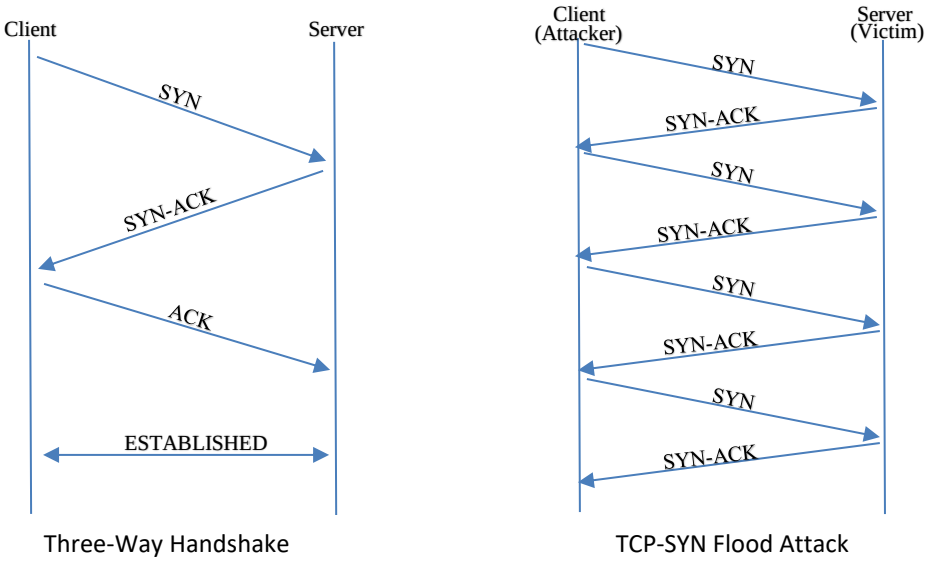


Figure 4. A comparison of TCP-SYN Flood Attack with a successful Three-Way Handshake.

HTTP-GET Flood

HTTP GET is the command used by the client to request a file from a web server. The purpose of an HTTP GET flood attack is to reduce the number of HTTP connections that the web server can keep open by consuming server resources, rendering the server unresponsive to legitimate HTTP GET requests. In this type of attack, the attacker sends too many HTTP GET requests to the target server. The attacker can send an HTTP GET request on each TCP connection or can send multiple HTTP GET requests for each TCP connection using the HTTP 1.1 feature. Moreover, these requests do not require high network traffic[22].

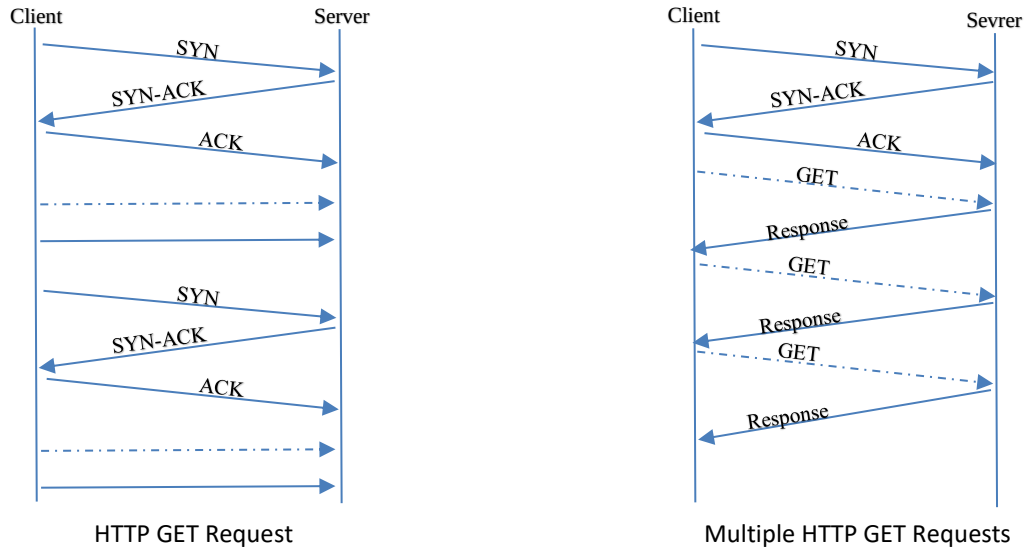


Figure 5. The comparison of singular and multiple HTTP GET requests in HTTP GET Flood attack [17]

2.3.2 PortScan

The PortScan attack in the CICIDS2017 dataset is made by the program named Nmap[23] and is located on Friday afternoon section [4]. Ports are abstract connection points named with numbers between 0 and 65535 and are used to determine what type of service you want to receive from the reached computer. PortScan is a popular attack type used to gather information about how attackers can get into the system. With this attack, the attacker can learn much critical information about the connected devices, such as operating system, running services, and port status. In a PortScan attack, the attacker sends a message to each port. The type of response received from the victim gives information about whether the port is used or not. By using this information, the attacker learns about the used system and detects its weaknesses[24].

Port Scanning types can be listed as follows:

SYN Scan: In this attack, numerous SYN flagged packets are sent to different ports of the attacking victim device, and the victim is expected to respond to these packets with SYN / ACK. When the attacker gets the response, he breaks the link and leaves incomplete the three-way handshake. In large quantities of packages with SYN flags from one computer is a symptom of this attack [25].

TCP Connect Scan: In this attack, the attacker attempts to establish a large number of TCP connections over different ports with the victim. After establishing the connection, the attacker who detects the open and closed ports terminates the connection. Numerous TCP connections from one computer through different ports point to this attack [25].

ACK Scan: In this attack, numerous ACK flagged packets are sent to different ports of the attacking victim device, and the victim is expected to respond to these packets with SYN / ACK. When the attacker gets the response, he breaks the link and leaves incomplete the three-way handshake. In large quantities of packages with ACK flags from one computer is a symptom of this attack [25].

FIN Scan: In this attack, a large number of packages with FIN flags are sent to different ports of the victim device from the attacker. While this message is ignored in the open ports, the closed ports respond with the RST message. Thus, the attacker learns about open and closed ports. In large quantities of packages with FIN flags from a computer point to this attack [25].

NULL Scan: In this attack, many packets with no flag are sent to different ports of the victim machine from the attacker. While this message is ignored in the open ports, the closed ports respond with the RST message. A large number of packages without flags from one computer indicate this attack [25].

XMAS Scan: In this attack, a large number of packages with FIN, PSH and URG flags are sent to different ports of the victim device. While this message is ignored in the open ports, the closed ports respond with the RST message. Thus, the attacker learns about open and closed ports. In large quantities of packages with FIN, PSH and URG flags from a computer point to this attack [25].

UDP Scan: In this attack, the attacker attempts to establish a large number of UDP (User Datagram Protocol) connections over the different ports with the victim [25].

Fragmentation Attack: In this attack, which is used to bypass the firewall, the attacker sends packets in small chunks. These parts can pass without being caught in the firewall. Too many packets containing very short string expressions point to this attack [25].

2.3.3 **DDoS**

The DDoS attack in the CICIDS2017 dataset is made by the program named LOIC [26] that sent HTTP, TCP and UDP requests. is located on Friday-afternoon section [4]. A DDoS (Distributed Denial-of-Service) attack is another type of attack designed to prevent a legitimate user from accessing a website or web service, such as a DoS attack. Unlike the DoS attack, the DDoS attack is not made from a central computer but is made from many computers to a single source [17].

Hundreds of proxy computers can be used to attack a computer. These computers used were infected by malicious software so that they could be used remotely during the DDoS attack. These devices are called "zombies", and the network created by these devices is called "botnet" [17].

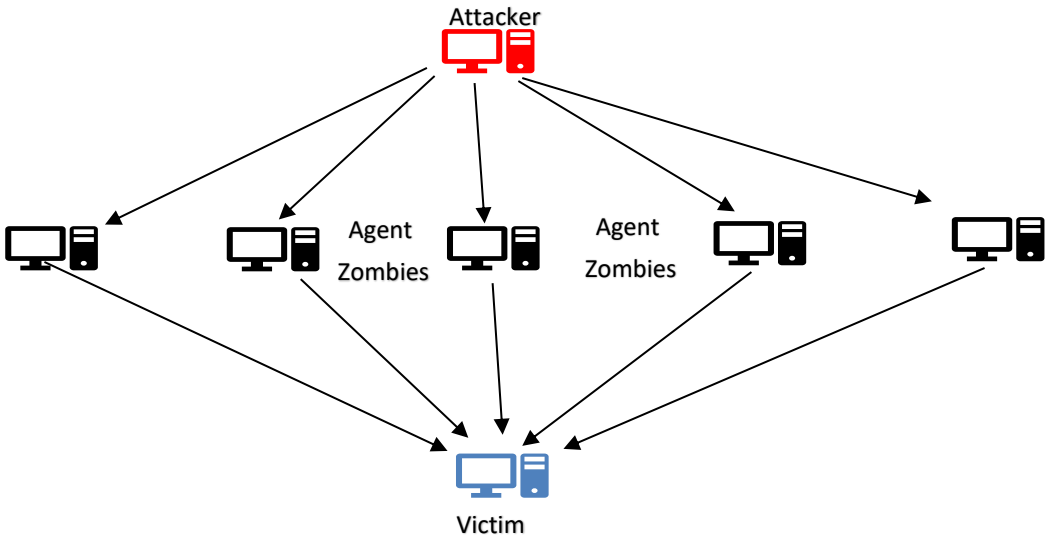


Figure 6. Representation of a DDoS Attack

2.3.4 **DoS Goldeneye**

DoS Goldeneye [27] takes place on Wednesday-records with other DoS attacks within the CICIDS2017 dataset [4]. It is a Python-based DoS attack. The intent of this attack is to consume the victim's system resources, thus preventing legitimate users from receiving service. Goldeneye is a multithreading attack, which can influential launch an http Flood attack using multithreaded CPU and memory hardware efficiently[20].

It does not encrypt packets during an attack and does not create a fake source IP (Internet Protocol) address (IP spoofing). It can be run on all Linux, Mac and Windows operating systems[20].

By using the Keep-Alive method, it maximizes the file size it transmits over a single TCP connection. Also, with the No-Cashed message feature, it disables HTTP- Cache-Control. By using these two features, system resources are consumed very quickly[20].

2.3.5 **FTP-Patator**

FTP-Patator attack is made using the Patator [28], a multithreaded tool written in the Python program. This attack takes place on Tuesday-morning-records within the CICIDS2017 dataset [4].
FTP (File Transfer Protocol) is a network protocol that provides file transfer between client and server in a network. To transfer a file using FTP, a valid username and password are required on the network to be sent. The FTP-Patator attack is a brute-force attack aimed at seizing this username and password[29].

Brute-force is a cryptographic attack designed to obtain passwords. In this attack, the attacker tries to log on to the system as a legitimate user by trying possible usernames and passwords on the system. Software that automatically generates passwords is often used for this process. Because users tend to prefer meaningful and rememberable passwords, passwords are generally weaker than they are thought to be against brute-force attacks[29].

A large number of unsuccessful entry attempts over a short period of time is a characteristic feature for brute force attacks. In this context, it is possible to see dense packet flow during brute-force attack. In addition, failed entries do not contain high sizes files, so bandwidth consumption and bytes count are low[29].

2.3.6 SSH-Patator

SSH-Patator attack is made using the Patator[28], a multithreaded tool written in the Python program. This attack takes place on Tuesday-afternoon-records within the CICIDS2017 dataset[4].

SSH (Secure Shell) is a cryptographic protocol that allows secure operation of various network services over a network in an unsecured environment (e.g. The Internet). The most common use of this protocol is to log on to a remote system[30].

In this attack, the intent of the attacker is to gain remote access to a system and gain full control over it. This attack consists of three steps[31]:

Scanning Step: This is the first phase of the SSH-Patator attack. The goal in this phase is to learn about the system to be attacked (See 2.2.2 PortScan). The attacker tries to find the host running SSH by performing a scan on a specific port number for IP blocks on a network or subnets.
Brute-Force Step: At this stage, the attacker tries to log in to the system he has discovered during the scanning phase with a large number of username and password combinations (See 2.2.5 FTP-Patator).

Die-off Step: The attacker gains the legitimate user authority by successfully logging in.

While during the first step of this attack, a large number of incomplete TCP packets with SYN flags are observed, in the second step (brute-force), small sized completed TCP packets are seen. The number of packets in the stream is high, but the packet sizes are low.

2.3.7 DoS Slowloris

DoS Slowloris [32] takes place on Wednesday-records with other DoS attacks within the CICIDS2017 dataset [4]. The intent of this attack is to consume the system resources of the victim and thereby prevent legitimate users from receiving service.

This attack, written in Perl programming language, can run on Windows and Linux operating systems. This attack creates a TCP-SYN flood (See DoS HULK) and aims to reduce the number of users that the server can serve [20]. The target web server is contacted in large quantities and it is targeted that these connections stay connected for as long as possible.

During this attack, a large number of incomplete TCP packets with SYN flags are observed. Package sizes are small, so bandwidth consumption and bytes count are low.

2.3.8 DoS SlowHTTPTest

DoS SlowHTTPTest [33] takes place on Wednesday-records with other DoS attacks within the CICIDS2017 dataset [4]. This attack abuses the window size feature of TCP, consuming the resources of the victim server so that the rightful users cannot benefit from the service.

The window size is a feature used to prevent overflow and stacking of data in TCP. In this way, the recipient limits the maximum size of the file be received and the server makes submissions based on this limitation. By this method, avoids the overflow and collapsing of the transmitted data due to the slowness of the connection during transmission[34].

During the SlowHTTPTest attack, the attacker sends a normal request to the server, but when receiving the response from the server, it sets the window size to a value too close to 0, which slows down the receiving process as much as possible. In such a case, the server will have to allocate resources to process and store the remaining large part after sending a very small part of the file. In the event of an increase of these requests, the server cannot meet the demands and becomes out of service. In addition to the attack, the attacker continues sending SYN and ACK packets to prevent the connection from breaking due to timeout[34].

This attack is quite easy to make because it does not require high bandwidth or hardware. Moreover, this attack is difficult to detect in the internet stream because it looks like innocent HTTP requests which need a long time [34]. However, connections that have window sizes that are much smaller than normal flow will give you a hint about this attack.

2.3.9 Botnet

Botnet attack is made using the Ares [35], an attack tool written in the Python program. This attack takes place on Friday-morning-records within the CICIDS2017 dataset[4].

Botnet is a network created by malware-infected computers. These computers that make up botnets are called bots or zombies. Because of the malicious software, these computers perform various harmful activities without knowledge of their owners. These computers can be remotely controlled by bot-masters and used to make SPAM (unsolicited emails) or DDoS attacks[36].

The detection of the bots' network behaviour is only possible in the active state (during the attack). When the botnet is passive, there is no network activity and cannot be detected. In the active state, the number of bytes per packet and the number of packets with PSH flag make it possible to detect this attack[36].

2.3.10 Web Attack

Web Attack takes place on Thursday-afternoon -records in the CICIDS2017 dataset [4]. In this group of attacks, three different web-based attacks were launched against a vulnerable PHP (Hypertext Pre-processor - a script programming language)/ MySQL (an open source database administration framework) web application identified as the victim. These attacks are:

Brute Force: This topic has been elaborated under the heading FTP-Patator.

XSS: XSS (Cross-Site Scripting), a popular web-based attack type, is implemented by injecting code into a web page. When the victim wants to view this web page, this malicious code fragment can lead to undesirable results such as data theft, session seizing, special code execution, and fraud [37].

SQL Injection: SQL (Structured Query Language – a database management system) Injection attack is one of the most popular and most dangerous attacks on the web. A typical dynamic web page keeps the user's various information in the database for later use and occasionally makes various queries in the database to reach this data [17].

These attacks are usually done by exploiting a security vulnerability in the database layer of a web application. The attacker can access the database using the exploits of the connection between the web application and the database. Then he could be able to steal, delete, or modify the data by sending malicious commands to the database server [17].

2.3.11 **Infiltration**

Infiltration attack takes place on Thursday-afternoon -records in the CICIDS2017 dataset. However, the infiltration concept used in this dataset is not a general attack, but rather a specialized attack concept for a specific scenario. In this situation, there is an information gathering attack on a network that has become vulnerable due to a virus file entering the system [4].

This attack scenario is as follows: the virus enters the system through a file downloaded from the Dropbox by the victim using Windows and a file copied from a USB flash drive by the victim using the Macintosh. At the next stage of the attack, the attacker exploits the vulnerability created by this virus to perform port scanning attacks on the network [4].

2.3.12 **Heartbleed**

This attack is made using the Heartleech[38], a Heartbleed exploit tool written in C programming language. This attack takes place on Wednesday - afternoon -records within the CICIDS2017 dataset[4]. Heartbleed is a crucial vulnerability in OpenSSL (an open source implementation of SSL and TLS protocols) and exploits from heartbeat, a library of TSL protocols [39].

The normal operation of heartbeat is as follows: The client transmits to the server a request consisting of random payload. the server replies this request with a packet containing the same payload [39].

During the attack, a specially created heartbeat request is sent to the server. This demand is actually empty but points out that it carries very high amounts of data. At this stage, due to a vulnerability in OpenSSL, the server sends a piece of memory in the specified length in response to this message. These parts contain various confidential information such as personal information, usernames and passwords that should not be sent. The size of these packages can be up to 16 KB and the attacker can repeat this operation without any limit [39].

During this attack, some anomalies on the network flow are observed. The length of the incoming messages is less than the heartbeat minimum message size of 20 bytes. Outgoing

message lengths are in the kilobyte level, and for a heartbeat response, this size is too big. The difference in size between outgoing and incoming messages is another way to understand the attack. In normal heartbeat messages, the lengths of the outgoing and incoming messages are the same. In case of attack, incoming message lengths are too small, outgoing message lengths are too large [39].

2.4 Machine Learning

Machine learning is a science and art that enables the programmed computers to learn from the data given to them [40]. In the machine learning process, computers can be trained on the data (training set) given to them, and they can show their performance on a different data (test set). In this way, the problem is solved by minimizing human intervention. Machine learning is heavily used in many places where classical methods are inefficient. Areas of use can be listed as follows [40]:

- It can interpret very large and complicated data.
- It can solve complex problems that traditional methods cannot find solutions.
- Machine Learning methods can find solutions to situations where existing solutions require too much external intervention/update without external intervention.
- It can work in variable environments. Machine learning methods can be applied to a new situation by examining the data.

Machine learning algorithms are divided into 4 groups according to whether the training data are labelled or not and according to the training supervision they have received. These are supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning [40].

Supervised learning: In this method, the training data had been correctly classified and labelled. For example, all flows in the dataset contain information (tags/labels) about their nature (such as normal or harmful). At the next stage, the test/prediction phase, these tags are compared to the results found by the algorithm, and the algorithm's success is calculated [40]. The performance of the method is high. However, supervised learning is costly because it used external service (e.g. manual tagging) for labelling. Examples of such algorithms are Decision Trees, K-Nearest Neighbours, and Random Forests.

Unsupervised learning: There is not a labelling process in this method. The algorithm divides the data into groups according to various properties and observes their association with each other. It is used extensively in areas such as anomaly detection, relationship learning and dimension reduction [40]. Cost of this method is low because it does not require outsourcing expertise like labelling.

Semi-supervised learning: method is a hybrid method that occurs when supervised learning is combined with unsupervised learning method. Usually, a very small portion of the data is labelled, and the rest is unlabelled [40]. This method combines the high performance of supervised learning with the low cost of unsupervised learning.

Reinforcement Learning: this approach is very different from the other three as a basic principle. In this system, the algorithm receives a penalty for the wrong choices that are made during training and a reward for the right choices. So, the algorithm constructs its own rules [40].

In this project, supervised learning methods will be used to take advantage of having a manually-labelled good dataset. In this respect, it is aimed to achieve high-performance advantage without getting cost disadvantage.

Used machine learning methods used in the application phase are: Naive Bayes, QDA, Random Forest, ID3, AdaBoost, MLP, and K Nearest Neighbours. While choosing these methods, the focus is on bringing together popular algorithms with different characteristics. In this context, the algorithms used are examined in detail as follow.

2.4.1 **Naïve Bayes**

Naïve Bayes is a machine learning algorithm that is simplified with the addition of the independence condition on Bayes' theorem [41].

Bayes' theorem is expressed by the following equation:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \qquad \textbf{2.1}$$

P (A | B): The probability of occurrence of A in the case of B occurrences.
P (B | A): The probability of occurrence of B in the case of A occurrences.
P(A) and P(B): Prior probabilities of A and B.

Naive Bayes is a network of probabilities consisting of a parent node representing the unobserved state and multiple child nodes representing the observed states [41].

In this statistical network, it is assumed that the child nodes are independent of each other. Although this hypothesis is the basis of Naive Bayes' theory, the assumption that sub nodes are independent of one another is often not correct. This is the underlying reason for the Naive Bayes method to achieve lower accuracy when compared to other machine learning methods [41].

Despite this disadvantage, it is one of the most preferred machine learning methods because the training period is very short, and the computational cost is very low[41].

2.4.2 **Decision Trees**

Decision trees are one of the popular classifiers used in machine learning methods. In this approach, the rules used are fairly straightforward and understandable.

Each decision tree consists of nodes (root-node and sub-nodes), branches and leaves. Within each node, there is a decision statement. According to the result of this decision, the algorithm chooses one of the two branches in the next step (the number of branches may be more than

two in some sub-algorithms). This selected branch takes the algorithm to the next node. This process ends with the last element, the leaf [41].

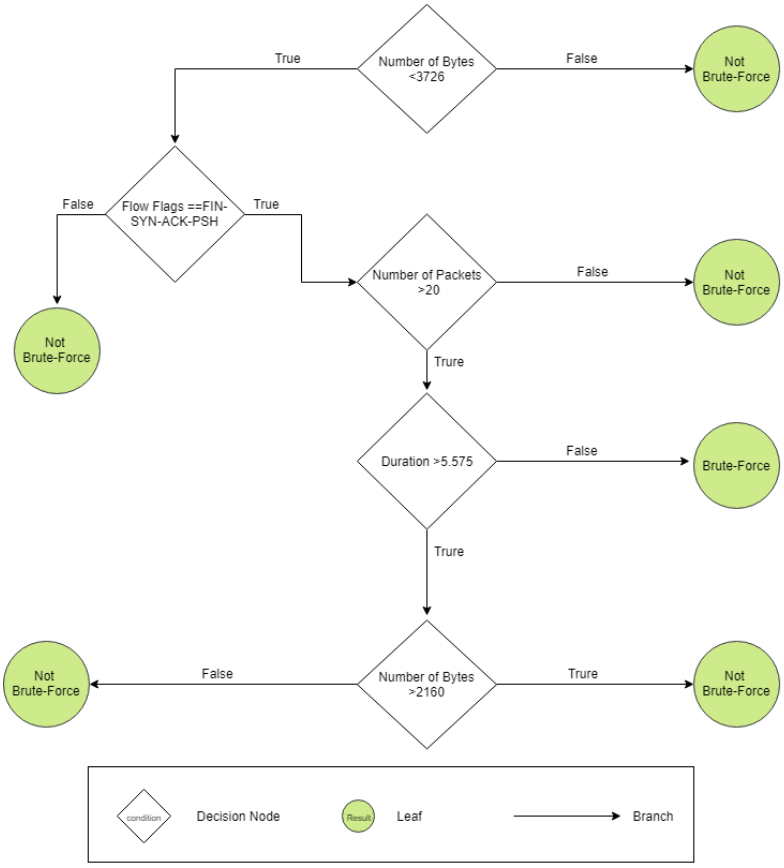


Figure 7. Detection of brute-force attack using decision trees [29].

Decision trees method applies the divide-and-conquer strategy. It makes very large and meaningless data smaller and group them into a meaningful one. It is therefore widely used for the classification of large and complex data[41].

Unfortunately, these complex trees lead to overfitting[2]and prevent getting fair results. Another disadvantage is that a slight misclassification of the data at the beginning of decomposition can lead to different branches and misleading results. In order to cope with this, usually, the data to be used must go through pre-processing.

In the implementation phase, a decision tree algorithm, ID3(Iterative Dichotomiser 3), is used. This algorithm is suitable for situations where the training set contains many features. It also stands out with its remarkable features such as giving a reasonable value without doing too much computation, and connecting more than two branches to decision nodes[41].

[2] In this case, a very close and very specific analysis of a dataset results in the analysis being very successful for this dataset, but this analysis is invalid for future observations.

2.4.3 **Random Forest**

Random forest is a machine learning approach that uses decision trees. In this method, a "forest" is created by assembling a large number of different decision tree structures which are formed in different ways[42].

Random forest generates N decision trees from training set data. During this process, it randomly resamples the training set for each tree. Thus, n decision trees are obtained, each of which is different from the other. Finally, voting is performed by selecting new estimates from estimates made by N trees. The value with the highest rating is determined as the final value[42].

The random forest has many advantages. These can be listed as follows [43]:

- This algorithm can work well with very large and complicated datasets.
- The overfitting problem frequently encountered by decision trees is very rare in this algorithm.
- It can be applied to a lot of kind of machine learning problem.
- It is also good to deal with missing values in the data set. It replaces these lost values with their own created values.
- In addition, it calculates and uses the importance level of the variables when making the classification. Thanks to this, it is also used for feature selection in the machine learning.

On the other hand, the structure of Random forest is quite complicated because it is made up of many decision trees. Another disadvantage is that It is pretty difficult to understand its functioning.

2.4.4 **K Nearest Neighbour**

KNN (K Nearest Neighbour), which is a sample-based method, is one of the most used machine learning algorithms with its simple and fast structure. This algorithm depends on the assumption that the examples in a dataset will exist close to the examples with similar properties in another dataset [41].

In this context, KNN identifies the class of new data that is not classifiable by using training data of known class type. This determination is made by observing the nearest neighbours of the new sample, for which no classifications are specified [41].

In a plane with N properties, the number of neighbours to be looked at for an unclassified sample is specified by the number K. For the unknown sample, the distances to the neighbours are calculated and the smallest K numbers are chosen from these distance values. The most repeated property within the K values is assigned as the unknown instance property. In Figure 5, a visual is created for the values 2 and 5 of K in a 3-dimensional plane (N = 3).

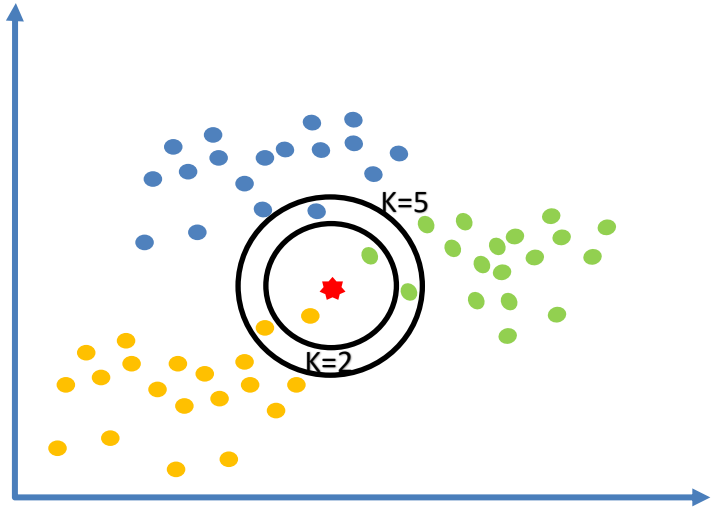


Figure 8. Operation of KNN algorithm for K = 2 and K = 5 values.

KNN, which provides good performance over multidimensional data and is a fast algorithm during the training phase, is relatively slow in the estimation stage [43].

2.4.5 **AdaBoost**

AdaBoost (Adaptive Boosting), a boosting method, is a machine learning algorithm developed to improve classification performance. The basic working principle of Boosting algorithms can be explained as follows: The data are first divided into groups with rough draft rules. Whenever the algorithm is run, new rules are added to this rough draft rules. In this way, many weak and low performance rules called "basic rules" are obtained [44].

Once the algorithm has been working many times, these weak rules are combined into a single rule that is much stronger and more successful. During this process, the Algorithm assigns a weighting coefficient to each weak rule, giving the highest coefficient value to the lowest error rate. These weight values come into play when final rules are selected. The final rule is created by giving priority to the high scored weak rules [44].

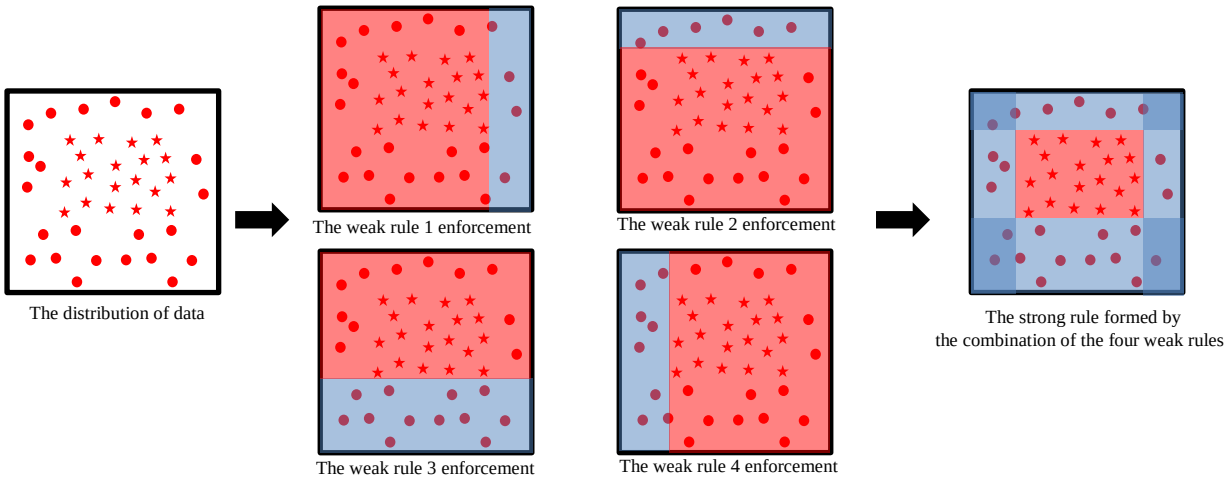


Figure 9. Demonstration of the operation of the AdaBoost algorithm.

AdaBoost algorithm has many advantages. These can be listed as follows [43]:

- There is no need for variable transformation to use it in this algorithm.
- It can perform operations on too many weak rules.
- The overfitting problem very rare in AdaBoost.
- It is also good to deal with missing values in the data set.

On the other hand, it can be mentioned as disadvantages that it is weak against noise and extreme values, and that the predictive value does not have the highest values when compared to other algorithms.

2.4.6 **MLP**

MLP (Multi-Layer Perceptron) is a genre of artificial neural networks. Artificial neural networks (ANN) is a machine learning method that takes inspiration from the way the human brain works. The intention of this method is to imitate the properties of the human brain, such as learning, decision making, and deriving new information. While the human brain is made up of interconnected cells called neurons, artificial neural networks are made up of interconnected hierarchical artificial cells[45].

MLP consists of three stages. These stages are the Input layer, the hidden layer, and the output layer. The input layer is the stage of the MLP that is responsible for receiving data. No information processing is performed on this layer. Only the received information is transmitted to the next layer, the hidden layer. Each neuron in this step bonds with all the neurons in the hidden layer[45].

In the hidden layer, the data sent from the input layer is processed and transmitted to the next layer, the output layer. In MLP, the number of hidden layers or the number of artificial neurons in this layer may change. For example, the MLP used in this study has a 3-step hidden layer and 13 neurons in each layer. The change in the layer count and neuron number directly affects MLP performance and yield. By increasing the number of layers and neurons, MLP can deal with more complex problems. However, the increase in the number of these elements will also lead to an increase in the duration of the algorithm[45].

In the output layer, which is the last layer, each cell is tied to all the cells in the hidden layer, and the results of the processed data in the hidden layer are served at this stage.

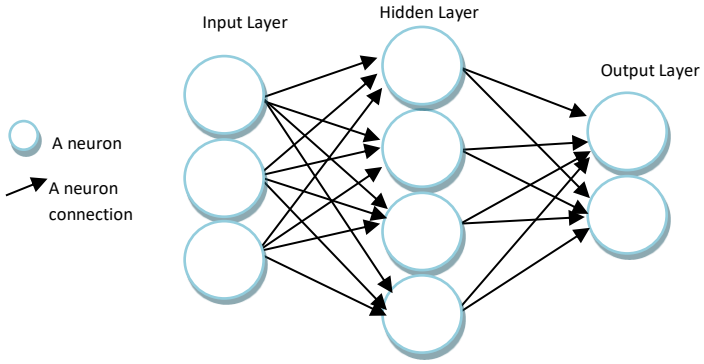


Figure 10. Demonstration of a three-layer MLP

The advantages MLP provides are as follows[43, 45]:

- It is good at coping with complicated problems.
- It can work with missing data.
- It can generalize after the learning process. Thus, it serves a wider area than the other machine learning algorithms.

But from the other side:

- It is difficult to build the network structure.
- The user should decide the appropriate network structure.
- Overfitting problems may be encountered.
- It is difficult to interpret and understand.

2.4.7 **QDA**

QDA (Quadratic Discriminant Analysis) is a discriminant analysis method. Discriminant Analysis is a statistical technique for assigning a measured data to one group among many groups. When this assignment is made, the observed data must be assigned to the group to which it belongs. If it is assigned a group that does not belong to it, an error occurs [46].

This mistake is known as "the error rate". The purpose of discriminant analysis is to perform the assignment process with a minimum error rate. Assuming that the data group has normal distribution and the variances are equal, the analysis is called Linear Discriminant Analysis. However, if the assumption of equality of groups' variances is not correct, a Quadratic Discriminant Analysis is obtained [46].

In order to be able to apply the Quadratic Discriminant Analysis, the number of samples observed must be greater than the number of groups [46].

2.5 **Related Work**

In this section, various studies using machine learning to detect anomalies on computer networks have been examined chronologically. In each study, the used machine learning algorithms, datasets and performance ratios are given. When selecting these studies have focused on the use of different machine learning algorithms and datasets.

In the study conducted in 2005 by Chebrolu et al[47], a hybrid structure was obtained by combining Bayesian networks (BN) and Classification and Regression Trees (CART) methods in order to produce a more efficient detection process by reducing the number of features examined during the detection of attacks. Using this method on the KDD cup 99 intrusion detection dataset, the following accuracy rates were obtained: Normal (Benign): 100%, Probe: 100%, DOS: 100%, U2R: 84% and R2L: 84%.

In 2007, another study[48], a high accuracy rate was obtained (98%, 88%, and 84%, respectively) for the Normal, Probe, and DOS tags using the Support Vector Machines (SVM) method on the DARPA 98 dataset. However, the accuracy rate of U2R and R2L attacks is less than 20% (0%, 18% respectively). This work is combined with the Dynamically Growing Self-

Organizing Tree (DGSOT) algorithm to enhance the effectiveness of the SVM algorithm. a significant increase in accuracy rates has been observed as a result of this process (Normal: 95%, DOS: 97%, U2R: 23%, R2L : 43%, and Probe : 91%).

A study targeting DDoS attacks was conducted by Suresh and Anitha [49], in 2011. In this study, CAIDA was chosen as the dataset while many machine learning algorithms such as Fuzzy C-Means, Naive Bayesian, SVM, K-Nearest Neighbours (KNN), Decision Tree and K-means Clustering were used. The success rates of this study are as follows: Fuzzy C Means: 98.7%, Naive Bayesian: 97.2%, SVM: 96.4%, KNN: 96.6%, Decision Tree: 95.6%, and K-Means Clustering: 96.7%.

In the study[50] conducted in 2012, Naive Bayes Classifier, combined with feature reduction method, achieved high performance in determining four types of attack on NSL-KDD dataset. (DoS: 98.7%, Probe: 98.8%, R2L: 96.1%, U2R: 64%).

In 2013, another study[51], a new architecture was achieved by combining K-Means Clustering and Naïve Bayes Classifier methods to deal with common false alarm, (the benign data classified as attack, false positive) problems in machine learning methods. In this study, the new method was tested with the ISCX 2012 Intrusion Detection Evaluation Dataset. At the end of this process, high performance (99.8%) and low false alarm rate (0.13%) were obtained.

Finally, in a study [4]conducted in 2017, seven commonly used machine learning methods (Naive-Bayes (NB), Random Forest (RF), K- Nearest Neighbours (KNN), Multilayer perceptron, Adaboost, ID3, and Quadratic Discriminant Analysis (QDA) were used to detect 15 different types of attack. During this process, CICIDS2017 was used as the dataset. The performance ratios obtained in this study are as follows: Naive-Bayes 0.84%, KNN 0.96%, RF 0.97%, MLP 0.76%, Adaboost 0.77%, ID3 0.98%, QDA 0.92%.

To summarize briefly, there have been many studies [4, 47-51] on network security with the anomaly-based network for more than 10 years. In these studies, network anomaly is detected by using many different data sets (DARPA98[5], KDD99[8], CAIDA[10], NSL-KDD[12], ISCX 2012[14] and CICIDS2017[16]) and machine learning algorithms (Naïve Bayes[4, 49, 50], KNN[4, 49], and Random Forest[4, 49] etc.).

The development of technology over time has led to many innovations in the network security area. New countermeasures and solutions are being produced against new attacks that are emerging every day. In this respect, this study will try to provide current solutions to current problems on network anomaly.

In order to carry out this purpose, this work follow the study done by Sharafaldin et al[4]. The reasons for this preference can be listed as follows.

- The work is quite recent, so it is compatible with today's technology.
- The data set used is current. A wide variety of protocols has been used to generate normal data.
- The dataset has a broad and up-to-date attack spectrum.

- The variety of machine learning algorithms is widespread. Among the studies examined, this work is one of the studies that use the most variety of machine learning methods.

3 Methodology

3.1 Tools and Methods

3.1.1 Software Platform

Python[52], a free and open source object-oriented programming language, draws attention with its simple syntax and dynamic structure. In Python, it's very easy to write code and analyse code. Another advantage is that it has the advantage of extensive documentation (books, internet sites, forums, etc.). In addition to all these advantages, it works in concert with many libraries which "machine learning" applications can be done. In this context, Python3.6 has been chosen to be used in this work, because of many of the advantages it provides.

Sklearn[53] (Scikit-learn) is a machine learning library that can be used with the Python programming language. Sklearn offers a wide range of options to the user with its numerous machine learning algorithms. Sklearn has extensive documentation and contains all the algorithms needed for this work.

Pandas[54] is a powerful data analysis library running on Python. When working with a large dataset, Pandas allows you to easily perform many operations such as filtering, bulk column / row deletion, addition, and replacement. Because of all these advantages, the Pandas library has been used.

Matplotlib[55] is a library that runs on Python, allowing visualization of data. This library is used to create graphs used in the study.

NumPy[56], a Python library that allows you to perform mathematical and logical operations quickly and easily, has been used in calculations in this work.

3.1.2 Hardware Platform

An evaluation criterion for machine learning algorithms is the execution time. However, the execution time may vary depending on the performance of the computer being used. Because of this, the technical specifications of the computer used in the application are shared. The technical characteristics of the computer used in the implementation phase are:

Central Processing Unit	:	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz
Random Access Memory	:	8 GB (7.74 GB usable)
Operating System	:	Windows 10 Pro 64-bit
Graphics Processing Unit	:	AMD Readon (TM) 530

3.1.3 Performance Evaluation Methods

The results of this study are evaluated according to four criteria, namely accuracy, precision, f-measure, and recall. All these criteria take a value between 0 and 1. When it approaches 1, the performance increases, while when it approaches 0, it decreases.

Accuracy: The ratio of successfully categorized data to total data [57].

$$\text{Accuracy} = \frac{TN+TP}{FP+TN+TP+FN} \quad \textbf{(3.1)}$$

Recall (Sensitivity): The ratio of data classified as an attack to all attack data [57].

$$Recall = \frac{TP}{TP+FN} \quad \textbf{(3.2)}$$

Precision: The ratio of successful classified data as the attack to all data classified as the attack [57].

$$Precision = \frac{TP}{FP+TP} \quad \textbf{(3.3)}$$

F-measure (F-score/F1-score): The harmonic-mean of sensitivity and precision. This concept is used to express the overall success[57]. so, in this study, when analysing the results, it will be focused, especially on the F1 Score.

$$\text{F- measure} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} \quad \textbf{(3.4)}$$

In calculating these four items, the four values summarized below are used:

- **TP**: True Positive (Correct Detection) The attack data classified as attack
- **FP**: False Positive (Type-1 Error) The benign data classified as attack.
- **FN**: False Negative (Type-2 Error) The attack data classified as benign.
- **TN**: True Negative (Correct Rejection) The benign data classified as benign.

This distribution is presented by visualizing Confusion matrix in Figure 11, also.

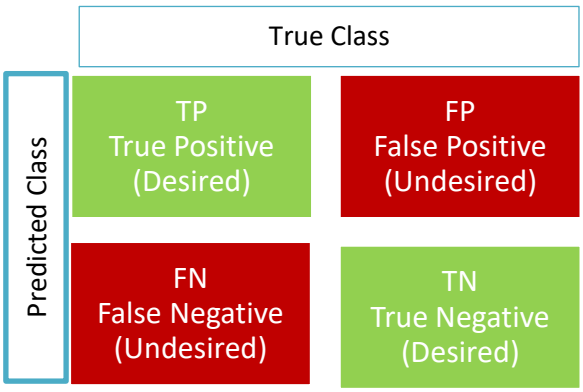


Figure 11. A confusion matrix

In addition to these 4 measures, it was included in this list, considering that the processing time is also an important factor in selecting algorithms, although it is not considered a success criterion.

3.2 Implementation

In this section, various pre-processing and actual application are performed to detect anomaly by machine learning techniques. For this purpose, the data cleansing process is performed in the first step and the dataset is cleaned from mistakes and defects. Then, the data set is divided into two parts, training, and test. After these operations, the properties to be used by the algorithms are decided at the step of feature selection. Finally, the section ends with the implementation of machine learning algorithms. Figure 12 illustrates the implementation process in detail.

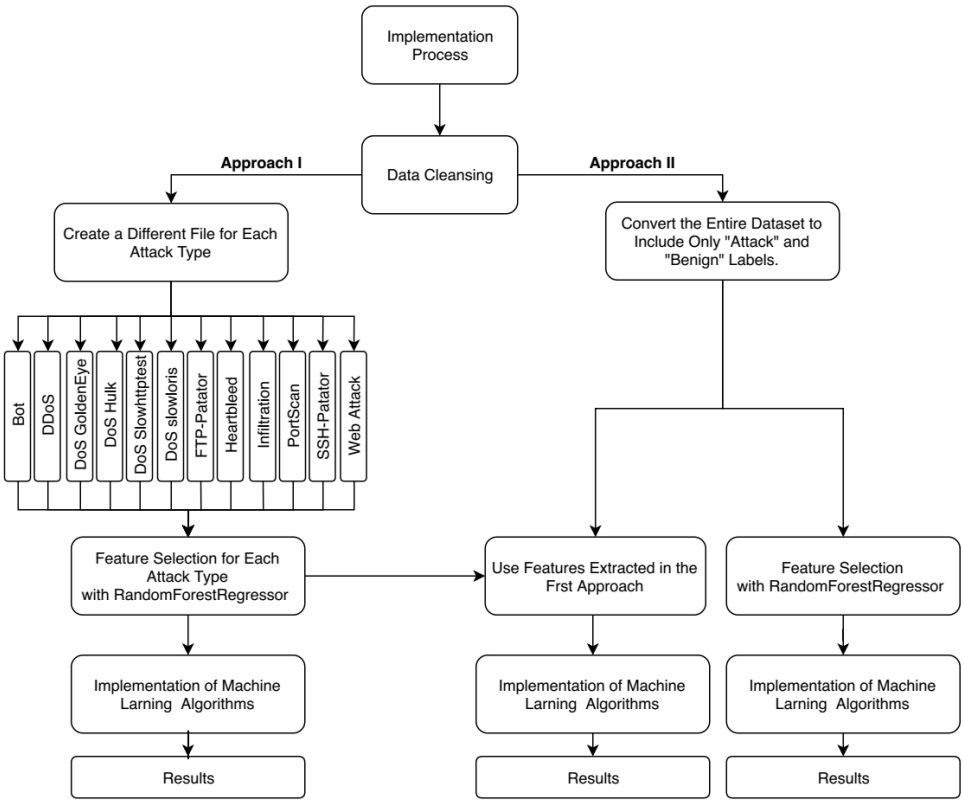


Figure 12. The Implementation Process

3.2.1 Data Cleansing

It may be necessary to make some changes to the dataset before using it in practice, making it more efficient. For this purpose, in this section, some defects of the CICIDS2017 dataset are corrected, and some data are edited.

The dataset file contains 3119345 stream records. The distribution of these stream records can be seen from Table 2. When these records are examined, it can be seen that the 288602 record is incorrect / incomplete[3]. The first step in the pre-processing process will be to delete these unnecessary records.

Label Name	**Number**
Benign	2359289
Faulty	288602
DoS Hulk	231073
PortScan	158930
DDoS	41835
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack – Brute Force	1507
Web Attack – XSS	652
Infiltration	36
Web Attack – SQL Injection	21
Heartbleed	11

Table 2. Distribution of stream records in the CICIDS2017 dataset.

Another error about the dataset is in the columns that make up the features. The dataset file consists of 86 columns that define the flow properties such as Flow ID, Source IP, Source Port etc. However, the `Fwd Header Length` feature (which defines the forward direction data flow for total bytes used) was written two times (41st and 62nd columns). This error is corrected by deleting the repeating column (column 62).

Another change that needs to be made in the dataset is to convert the properties including the categorical and string values (Flow ID, Source IP, Destination IP, Timestamp, External IP) into numerical data to be used in machine learning algorithms. This can be done with `LabelEncoder()`[58] from Sklearn classes. In this way, various string values that cannot be used in machine learning operations will get integer values between 0 and n-1 and will become more suitable for processing.

[3] The incomplete flow is a record saved as: ",,,"

However, although the “Label” tag is a categorical feature, no changes have been made on it. The reason is that during the processing, the original categories are needed in order to classify the attack types in different forms and to try different approaches.

Finally, some minor structural changes should be made to the dataset, including:

- In the Label feature, the character "–" (Unicode Decimal Code &#8211) used to identify the web attack subtypes (Web Attack - Brute Force, Web Attack - XSS, Web Attack - SQL Injection) must be replaced with the character "-” (Unicode Decimal Code &#45), since utf-8, the default codec of Pandas library, does not recognize it. Otherwise, the Pandas library that will not recognize this character and it will fail.

- "Flow Bytes/s", "Flow Packets/s" features include the values "Infinity" and "NaN" in addition to the numerical values, which can be modified to -1 and 0 respectively to make them suitable for machine learning algorithms.

3.2.2 Creation of Training and Test Data

During the machine learning process, data is needed so that learning can take place. The data sets used are the result of this need. In addition to the data required for training, test data is needed to evaluate the performance of the algorithm and to see how well it works. The algorithm acquires a skill on the training data and applies it to the test data. The result of the test data is the performance of the machine learning algorithm [40].

However, the CICIDS2017 dataset used does not contain dedicated training and test data, but it contains a single unbundled dataset. Therefore, the data should be divided into training and test data parts. In the application phase, a Sklearn command, train_test_split[59] is used. This command divides the data into 2 parts at the sizes specified by the user. Generally preferred partitioning is 20% test, 80% training data [40] and this ratio is also preferred in this application. The train_test_split command makes the selection random when creating data groups. This process is known as cross-validation. In order to ensure that the results obtained during the application are solid, the creation of the training and the test data have been performed 10 times in succession. The results obtained are the arithmetic mean of the repeated operations.

3.2.3 Feature Selection

In this section, the features in the dataset are evaluated to determine which features are important to define which attack. The list of features and a detailed description are provided in the Appendix A.

3.2.3.1 Feature Selection According to Attack Types

To do this calculation, a special file is created for every kind of attack by isolating the attack from other attacks. This file contains the entire stream identified as the attack and the data stream identified as randomly selected "Benign" (30% Attack, 70% Benign).

The Random Forest Regressor[58] class of Sklearn is used when importance weights of features are calculated. This algorithm creates a decision-forest. In this decision forest, each feature is given a weight of importance as to how useful they are in the construction of the decision-tree. When the process is finished, these importance weights of features are compared and sorted [60]. The sum of the importance weights of all the properties gives the total importance weight of the decision tree. The comparison of the score of any feature to the score of the whole tree gives information about the importance of that feature in the decision tree.

However, 8 features (Flow ID, Source IP, Source Port, Destination IP, Destination Port, Protocol, Timestamp, External IP) between 85 properties must not be included in the calculation, when the weight of importance is calculated. Although these features are used in classical approaches, it is possible that an attacker would prefer not to use well-known ports to escape control or to circumvent operating system constraints or he can use generated / fake IP addresses. Also, many ports are used dynamically, and many applications are transmitted over the same port. So, it can be misleading to use the port number [61].

In this context, while choosing the attribute importance of the attack, it will be much more effective to eliminate the misleading features such as IP address, Port number, Timestamp, use more generic and invariant attributes to define the attack. Because the shape of the data will give much more information about whether or not it is an attack.

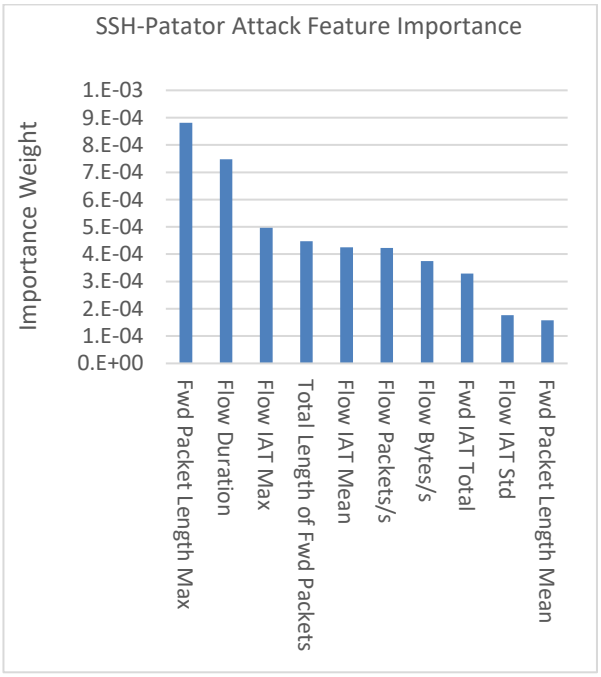
The distribution of features and four attributes with the most significance value for each attack can be seen from Table 3.

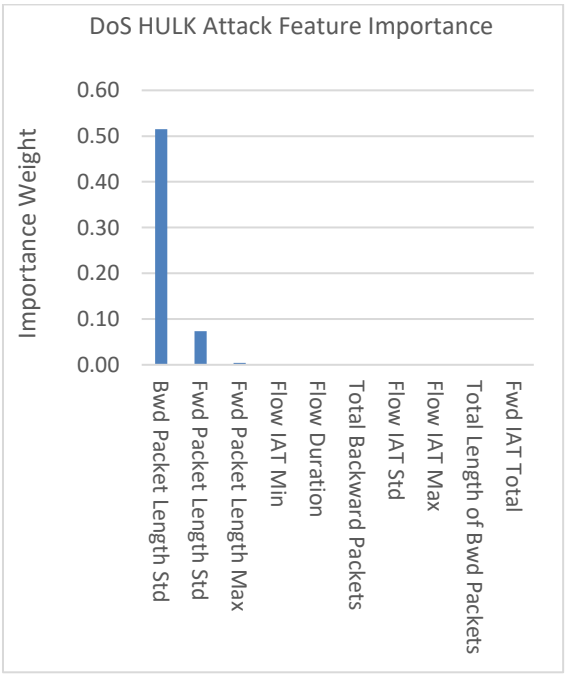
Attack / Feature Name	**Importance Weight**		**Attack / Feature Name**	**Importance Weight**
Bot			**FTP-Patator**	
Bwd Packet Length Mean	0.304823		Fwd Packet Length Max	0.063671
Flow IAT Max	0.034495		Fwd Packet Length Std	0.022751
Flow IAT Std	0.019464		Fwd Packet Length Mean	0.002179
Flow Duration	0.010129		Total Length of Bwd Packets	0.000746
DDoS			**Heartbleed**	
Bwd Packet Length Std	0.468089		Bwd Packet Length Mean	0.064
Total Backward Packets	0.094926		Total Length of Bwd Packets	0.056
Fwd IAT Total	0.012066		Flow IAT Min	0.056
Total Length of Fwd Packets	0.006438		Bwd Packet Length Std	0.044
DoS GoldenEye			**Infiltration**	
Flow IAT Max	0.442727		Total Length of Fwd Packets	0.05238
Bwd Packet Length Std	0.091185		Flow IAT Max	0.036096
Flow IAT Min	0.053795		Flow Duration	0.016453
Total Backward Packets	0.041583		Flow IAT Min	0.015448
DoS Hulk			**PortScan**	
Bwd Packet Length Std	0.514306		Flow Bytes/s	0.313402
Fwd Packet Length Std	0.069838		Total Length of Fwd Packets	0.304917
Fwd Packet Length Max	0.008542		Flow Duration	0.000485
Flow IAT Min	0.001716		Fwd Packet Length Max	0.00013

DoS Slowhttptest			**SSH-Patator**	
Flow IAT Mean	0.64206		Flow Bytes/s	0.000846
Fwd Packet Length Min	0.075942		Total Length of Fwd Packets	0.000814
Fwd Packet Length Std	0.022194		Fwd Packet Length Max	0.000749
Bwd Packet Length Mean	0.020857		Flow IAT Mean	0.000734
DoS slowloris			**Web Attack**	
Flow IAT Mean	0.465561		Total Length of Fwd Packets	0.014697
Bwd Packet Length Mean	0.075633		Bwd Packet Length Std	0.00536
Total Length of Bwd Packets	0.049808		Flow Bytes/s	0.00257
Total Fwd Packets	0.01868		Bwd Packet Length Max	0.001922

Table 3. The distribution of features and four attributes with the most significant value for each attack.

When the distribution of properties is examined, it appears that there are one or two features that stand out for almost all attack types. On the other hand, the characteristics of the Heartbleed and SSH-Patator attacks are quite different. For these attacks, there are a few features that have values that are very close to one another, not one or two dominant features. (See Figure13, the graphics and values of all features are provided in the Appendix B.)





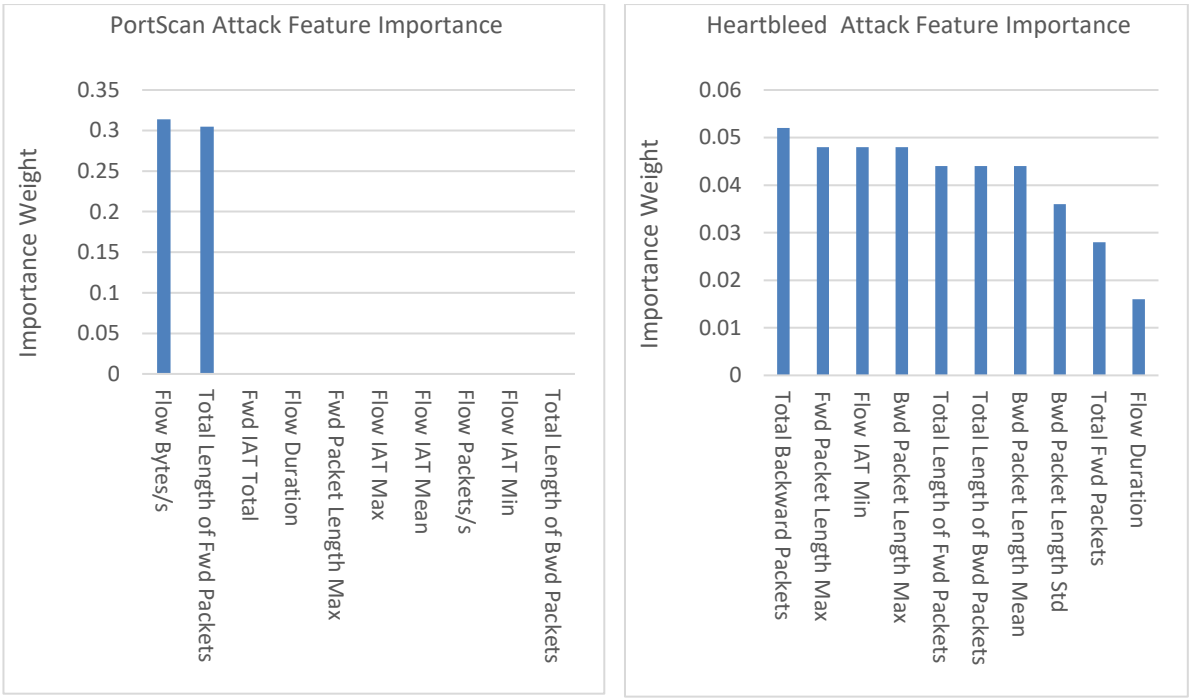


Figure 13. Graphs of feature importance weights of SSH-Patator, Heartbleed, DoS HULK, and PortScan attacks

For example, in the PortScan attack, two features ("`Flow Bytes/s` "- Number of flow bytes per second, "`Total Length of Fwd Packets`" - Total size of the packet in the forward direction) stand out. When the PortScan attack is examined, the attacker's behaviour is usually to send as many packets as possible with no payloads (only 20 Bytes transport layer header and 20 Bytes Internet layer header, 40 bytes in total) or with very few payloads [62]. Thus, the attacker both accelerates the process and makes the attack more effective and does not consume his bandwidth unnecessarily.

In this context, it is expected that the "`Total Length of Fwd Packets`" feature should be very low when it compared to benign flows. The data in Figure 14, are such as to support this prediction.

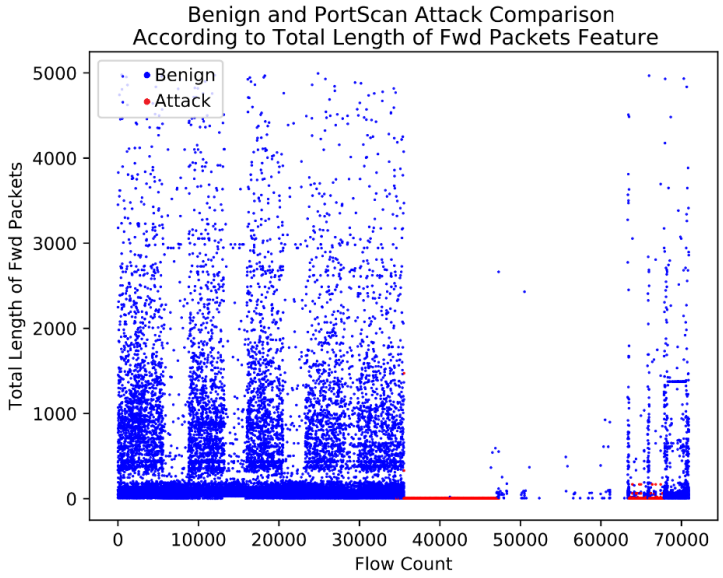


Figure 14. Benign and PortScan Attack Comparison According to Total Length of Fwd Packets Feature.

On the other hand, the SSH-Patator attack does not reveal any obvious feature. The importance values are very close to each other. The most important reason for this is that the SSH-Patator attack does not consist of a single step but a complex structure with three steps (Scanning Phase, Brute-Force Phase and Die-off Phase)[31]. SSH-Patator attack contains both PortScan (Scanning/Probe) and Brute-Force attacks. In this context, it is possible to see traces of both attacks in Figure 15.

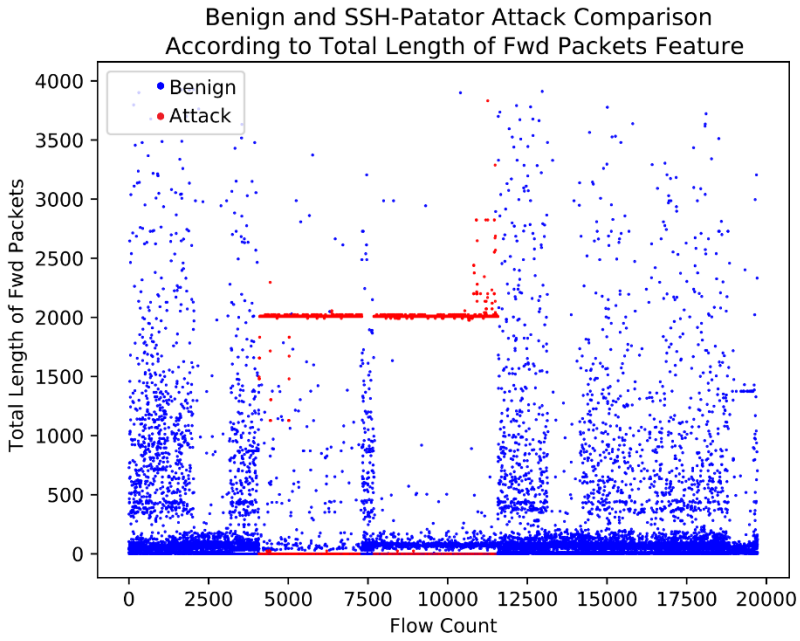


Figure 15. Benign and SSH-Patator Attack Comparison According to Total Length of Fwd Packets Feature.

3.2.3.2 Feature Selection According to Attack or Benign

Another approach in feature selection is to apply the Random Forest Regressor operation to the whole dataset by collecting all attack types under a single label; "attack". So, the data in this file contains only the attack and benign tags. As a result of this operation, the feature list obtained is shown in Table 4 and graphics of feature in Figure 16.

Feature Name	**Importance Weight**		**Feature Name**	**Importance Weight**
Bwd Packet Length Std	0.246627		Flow IAT Mean	0.003266
Flow Bytes/s	0.178777		Total Length of Bwd Packets	0.001305
Total Length of Fwd Packets	0.102417		Fwd Packet Length Min	0.000670
Fwd Packet Length Std	0.063889		Bwd Packet Length Mean	0.000582
Flow IAT Std	0.009898		Flow Packets/s	0.000541
Flow IAT Min	0.006946		Fwd Packet Length Mean	0.000526
Fwd IAT Total	0.005121		Total Backward Packets	0.000169
Flow Duration	0.004150		Total Fwd Packets	0.000138
Bwd Packet Length Max	0.004007		Fwd Packet Length Max	0.000125
Flow IAT Max	0.003579		Bwd Packet Length Min	0.000084

Table 4. According Attack and Benign Labels Feature Importance Weight List

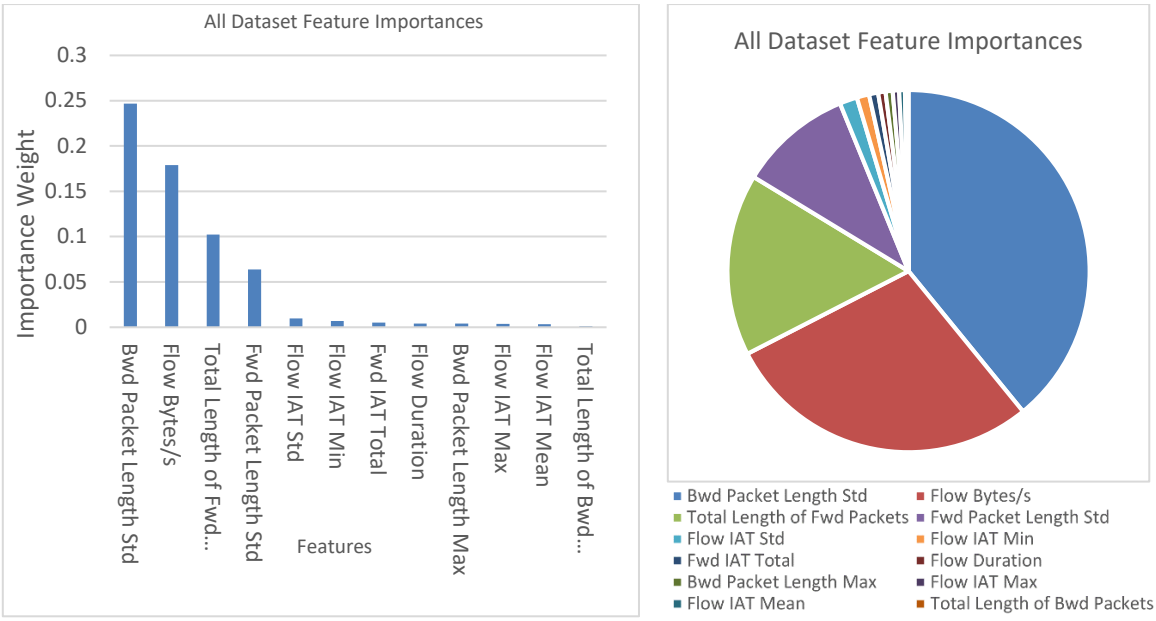


Figure 16. Graphs of Feature Importance Weights According to Attack and Benign Labels

3.2.4 Implementation of Machine Learning Algorithms

Two different approaches have been used to apply machine learning algorithms[4] to the dataset. In the first method, the files created in the Feature Selection section and the attributes obtained in the same section are used. These files contain 30% attack and 70% benign data and each of them named by the type of attack it contains.The seven machine learning methods are applied to each file 10 times, resulting in a separate outcome for each attack type. With this method, it is aimed to observe the effectiveness and performance of different machine learning methods on different attack types.

In the second approach, the entire data set is used as a single file. All attacks contained in this file are collected under a single common name; "attack". So, the data in this file contains only the attack and benign tags. The set of features to be used consists of combining the 4 features with the highest importance-weight achieved for each attack in approach 1 under a single roof. Thus, 4 features are obtained from each of the 12 attack types, resulting in a pool of features consisting of 48 attributes.After the repetitions are removed, the number of features is 18. The list of these features can be seen in Table 5

Bwd Packet Length Max	Flow IAT Mean	Fwd Packet Length Min
Bwd Packet Length Mean	Flow IAT Min	Fwd Packet Length Std
Bwd Packet Length Std	Flow IAT Std	Total Backward Packets
Flow Bytes/s	Fwd IAT Total	Total Fwd Packets
Flow Duration	Fwd Packet Length Max	Total Length of Bwd Packets
Flow IAT Max	Fwd Packet Length Mean	Total Length of Fwd Packets

Table 5. The feature list created for all attack types.

[4] Naïve Bayes, Random Forest, KNN, ID3, Adaboost, MLP, and QDA

An alternate way of implementing this approach is to use features with high weighting according to the importance scores obtained for the entire dataset in the Feature Selection According to Attack or Benign section, without using all of these eighteen features above. The threshold value for feature weight is set at 0.8%. In this way, 97% of the total feature importance weight will be covered by selecting only 7 features. The remaining 13 features constitute only 3% of the total weight of significance. If the features with a weight of 0.8% and above are selected, the following features are used:

Feature Name	Importance Weight	Percentage
Bwd Packet Length Std	0.246627	38.97%
Flow Bytes/s	0.178777	28.25%
Total Length of Fwd Packets	0.102417	16.18%
Fwd Packet Length Std	0.063889	10.10%
Flow IAT Std	0.009898	1.56%
Flow IAT Min	0.006946	1.10%
Fwd IAT Total	0.005121	0.8 %

Table 6. The importance weights obtained in "Feature Selection According to Attack or Benign" section.

4 Results and Discussion

In this section, the results of the studies done in the implementation section are presented. In this context, in the assessment carried out, the evaluation criteria are presented via the data of the F-measure. However, all the evaluation data obtained can be accessed from the Appendix.

The performance evaluation procedures are repeated 10 times for each machine learning algorithm. The numbers given in the tables are the arithmetic mean of these 10 processes. Box and whisker graphs are created to illustrate the consistency of the results and the change between them.

4.1 Approach 1 - Using 12 Attack Types

Seven different machine learning methods are applied to 12 different attack types and the obtained results are presented in Table 7.

Among the results presented in Table 7, the biggest and smallest achievements are emphasized as follows: The greatest scores are bold, the smallest scores are underlined and italics. In the results of the algorithms, if there is an equality in F-measure, the following values are examined in order to eliminate equality, respectively: accuracy, precision, recall, and time (In Table 7, only F-Measure values are given. See Appendix C for all values).

Attack Names	F-Measures						
	NB	RF	KNN	ID3	AB	MLP	QDA
Bot	*0.54*	0.96	0.95	0.96	**0.97**	0.64	0.68
DDoS	0.77	0.96	0.92	**0.96**	0.96	0.76	*0.34*
DoS GoldenEye	0.81	0.99	0.98	**0.99**	0.99	*0.64*	0.71
DoS Hulk	*0.23*	0.93	0.96	**0.96**	0.96	0.95	0.36
DoS Slowhttptest	*0.35*	0.98	0.99	0.98	**0.99**	0.78	0.38
DoS slowloris	*0.37*	0.95	0.95	**0.96**	0.95	0.74	0.46
FTP-Patator	**1.00**	1.00	1.00	1.00	1.00	1.00	1.00
Heartbleed	**1.00**	0.99	1.00	0.95	0.93	*0.66*	1.00
Infiltration	0.78	0.92	0.88	0.89	**0.92**	*0.52*	0.83
PortScan	*0.39*	1.00	1.00	**1.00**	1.00	0.61	0.85
SSH-Patator	*0.33*	0.96	0.95	**0.96**	0.96	0.83	0.41
Web Attack	0.74	0.97	0.93	**0.97**	0.97	*0.60*	0.84

Table 7. Distribution of results according to type of attack and machine learning algorithm.

When looking at the results, it is noticed that Random Forest, KNN, ID3 and Adaboost algorithms, have achieved over 90% success in detecting almost all attack types. Among these four algorithms, ID3, which is the most successful, has completed 7of 12 tasks with the highest score. In fact, in the 6 of these 7 tasks (DDoS, DoS GoldenEye, DoS Hulk, PortScan, SSH-Patator and Web Attack) ID3 shares the highest score with at least one algorithm. However, low processing time puts it ahead of other algorithms.

Naive Bayes, the algorithm with the lowest F-measure, is the last in 6 out of 12 tasks. However, it is also necessary to mention the QDA with the Naive Bayes here. Because in the almost all these 6 tasks, QDA has a very close score to Naive Bayes. It is known that Naive Bayes, a statistical method, has lower performance than other machine learning algorithms [41]. This assumption can also be extended to include the QDA, a statistical method. It is also noteworthy that the DoS Hulk, DoS Slowhttptest, and DoS Slowloris, which are the DoS resource depletion attacks, constitute half of these 6 tasks.

Another interesting point about Naïve Bayes is that it appears to have the highest score in the FTP-Patator attack. Although Naive Bayes was previously said to perform worse than the other algorithms, Naive Bayes is much better than the alternatives when it comes to speed. Naïve Bayes stands out with its speed because all the evaluation scores are equal to each other in this attack.

Another point of interest is that all machine learning algorithms achieve a full score in the FTP-Patator attack. This could be due to the fact that the features used to describe the FTP-Patator attack may become a characteristic that can be easily distinguished between normal and attack data, trapped in a very narrow range relative to normal data (See Figure 17).

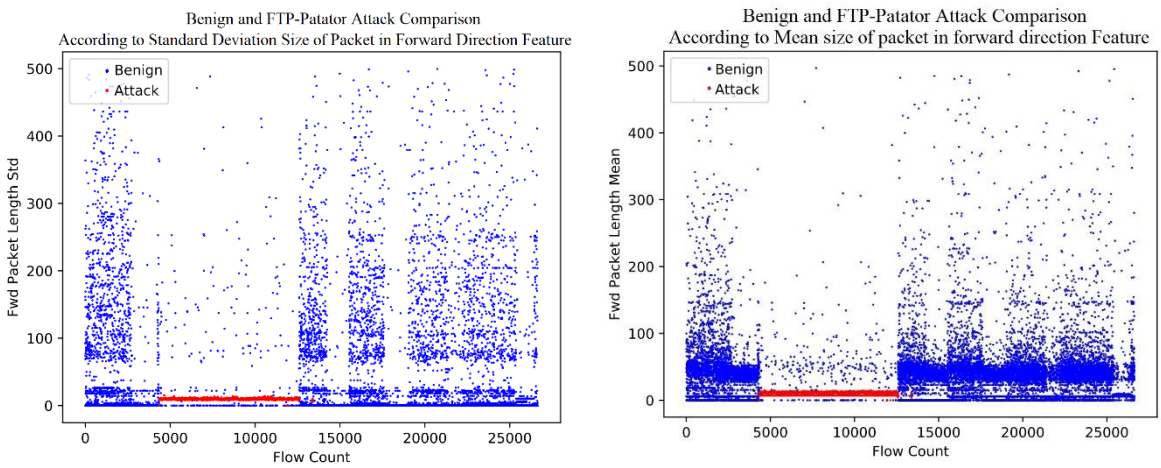


Figure 17. "Fwd Packet Length Std" and "Fwd Packet Length Mean" features in FTP-Patator and Benign flow.

MLP has the second worst performances among the algorithms. It has been the last in the 4 out of 12 tasks. Especially with the Heartbleed and Infiltration attacks, it has a fairly low score. three of the four attacks (Web Attack, Heartbleed, and Infiltration) with the low score have also smallest numbers of records in the dataset. In this context, it can be considered that there is a relationship between the achievement score of MLP and the amount of training data.

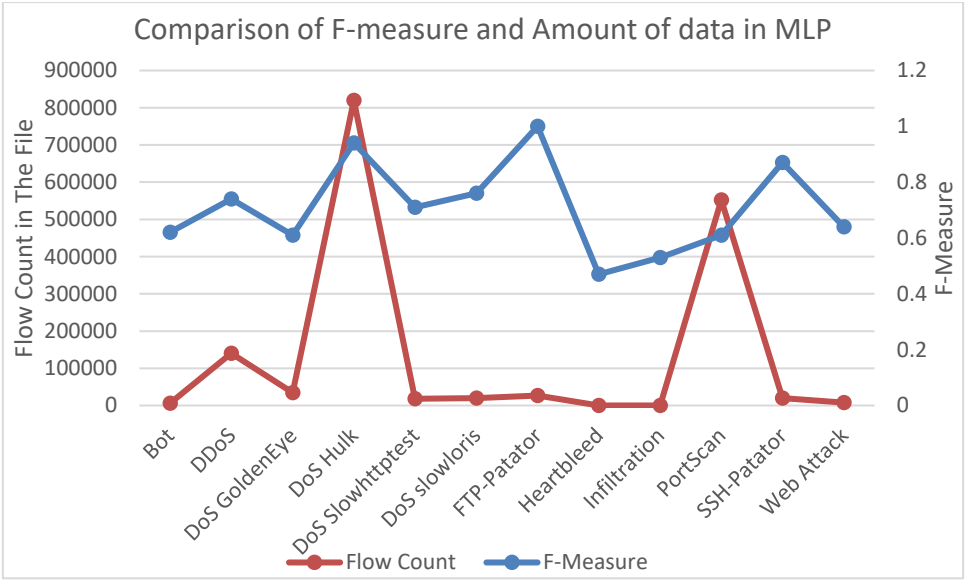


Figure 18. Comparison of the F-measures of the MLP and the flow numbers contained in the attack files.

Figure 18 shows a graph comparing the F-measures of the MLP algorithm and the flow numbers contained in the attack files. According to this graph, in the MLP algorithm, it can be seen that there is a correlation between performance and the amount of data. Based on this, it can be assumed that the increase in the amount of data in the MLP algorithm will contribute to the success of it. However, this assumption is not always verifiable.

There are other effects of not having enough data about Heartbleed and Infiltration attacks in the dataset. In order to better understand this effect, the F-measure data of any algorithm can

be examined. Figure 19 shows the box and whisker graphics obtained by applying the ID3 algorithm to nine attack files (See Appendix C for all graphics).

When these box and whisker graphics are examined, it appears that there is a balanced distribution for almost all attack types. The difference between the highest and lowest F-measure levels of all attack types is between 0%- 2%. However, this distribution is deteriorated in two types of attack (Infiltration and Heartbleed). The difference between the lowest and highest values in Heartbleed is 10%, while the difference between the highest and lowest values in Infiltration is 8%.

The reason for this unbalanced distribution is that there are not enough samples in the dataset for these two attacks (Infiltration is 36 flows and Heartbleed is only 11 flows in 2830743 flows).

This unbalanced distribution in the dataset unfortunately makes the results from these two attacks unreliable. Even though the results for these two attacks are very high, these figures are likely to be over-fitting. The large fluctuation between F-measure results also supports this assumption.

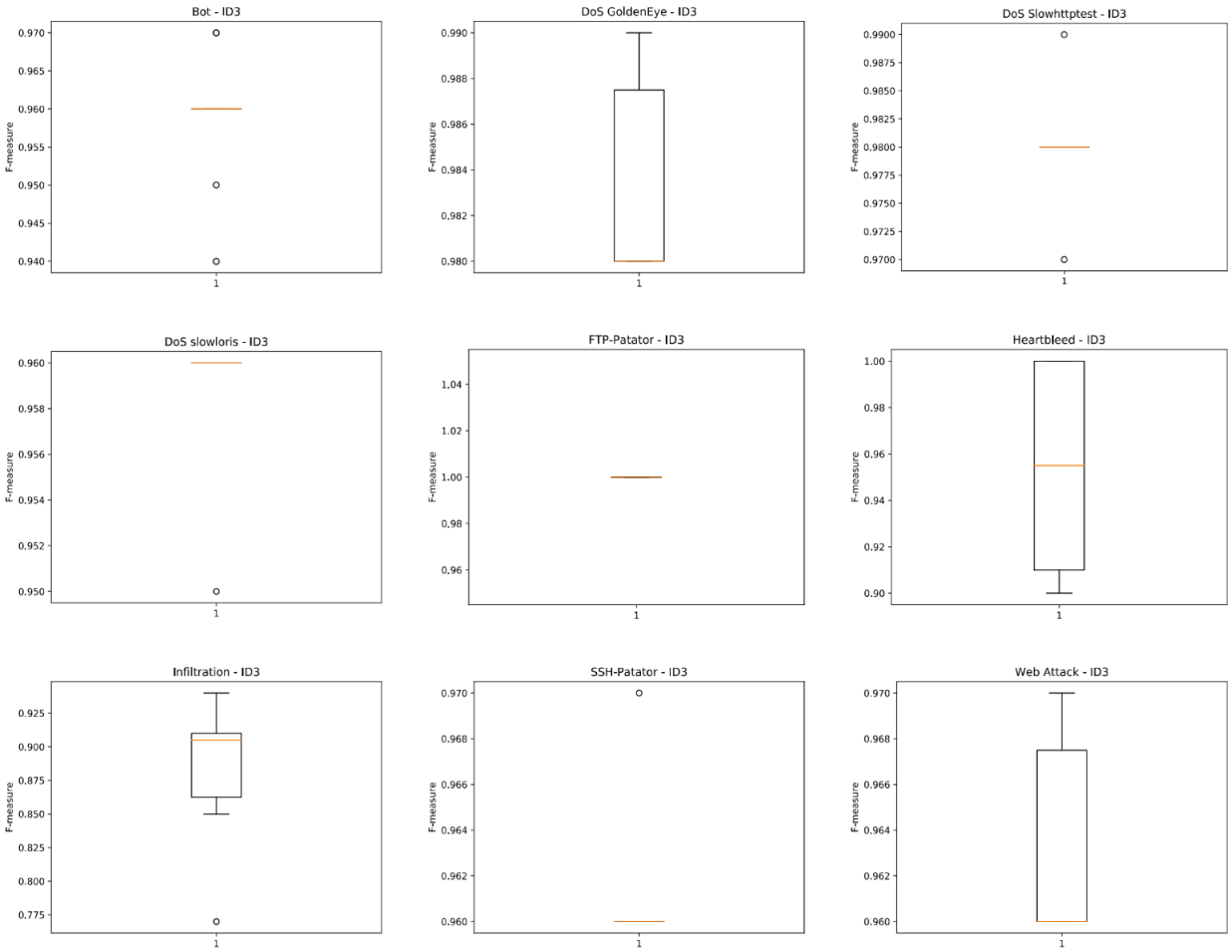


Figure 19. Box and whisker graphics containing the results of applying the ID3 algorithm to various attack types.

4.2 Approach 2 - Using Two Groups: Attack and Benign

In this section, the entire data set is used as a single dataset file. All attacks contained in this file are collected under a single common name, "attack". Seven different machine learning methods are applied to this dataset. In this approach, two methods will be used, the first one, the features created for attack files in approach 1 are used. In the second method, the 7 features obtained in the Feature Selection According to Attack or Benign section are used.

4.2.1 Using Features Extracted for Attacks Files

Table 8 shows the results obtained by using 18 properties (See second paragraph of Implementation of Machine Learning Algorithms section) extracted from the first approach. When Table 8 is examined, it is seen that the highest performance algorithm is the KNN with the score of 0.96. After that, the AdaBoost and ID3 come with 0.95. However, ID3 is remarkably faster than Adaboost, so it takes precedence with this feature. The lowest scoring algorithm is QDA with a score of 0.30. The QDA's score is about 0.40 points lower than the algorithms with the closest score (Naive Bayes and MLP).

From speed perspective, NB and QDA are the fastest algorithms. Although KNN has the highest performance score, it is noticeably slower than other algorithms.

Machine Learning Algorithms	**Evaluation Criteria**				
	F-Measure	Precision	Recall	Accuracy	Time[5]
Naive Bayes	0.79	0.80	0.78	0.78	**4.576**
QDA	*0.30*	0.84	0.31	0.31	6.649
Random Forest	0.94	0.95	0.94	0.94	24.739
ID3	0.95	0.95	0.95	0.95	29.284
AdaBoost	0.95	0.95	0.95	0.95	391.804
MLP	0.79	0.81	0.84	0.84	81.668
K Nearest Neighbours	**0.96**	0.96	0.97	0.97	*1967.054*

Table 8. Application of the features obtained in the first approach.

4.2.2 Using Features Selection for All Dataset.

An implementation that uses an alternative feature selection method can be seen in Table 9. In this method, the 7 features obtained in the Feature Selection According to Attack or Benign section are used. The changing parts are highlighted in red. When Table 8 and Table 9 are compared, there is no significant change in the algorithms of Random Forest, ID3, AdaBoost, and MLP based on F-measure. However, In Naive Bayes and QDA increase of 2 and 11 points were observed respectively.

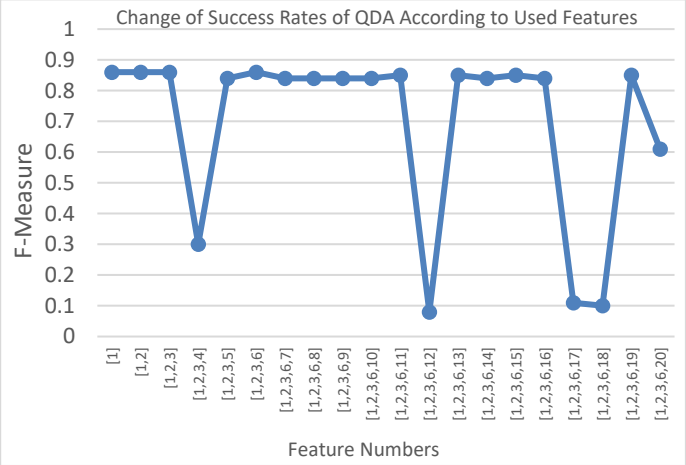
From speed perspective, the running times of all algorithms are noticeably reduced. The reason for this reduction in execution time is that 18 attributes are used in the method applied in Table 8, whereas only 7 attributes are used in Table 9. This reduction in feature count has reduced the running time of machine learning algorithms.

[5] in seconds.

Machine Learning Algorithms	Evaluation Criteria				
	F-Measure	Precision	Recall	Accuracy	Time
Naive Bayes	0.81	0.8	0.82	0.82	1.6258
QDA	*0.41*	0.83	0.38	0.38	1.925
Random Forest	0.94	0.947	0.94	0.94	20.511
ID3	0.95	0.95	0.95	0.95	11.552
AdaBoost	0.94	0.94	0.94	0.94	144.166
MLP	0.79	0.815	0.84	0.84	51.799
K Nearest Neighbours	0.97	0.97	0.97	0.97	1038.253

Table 9. Implementation of features obtained using Random Forest Regressor for All Dataset.

When looking at the table 9, it is seen that almost all of the results have F-measures above 0.80, but the score obtained in the QDA algorithm is well below this value. In order to investigate the cause of this problem, it can be seen in the figure 20 that illustrates how the QDA algorithm responds to the possibilities of different feature selection.



1-Bwd Packet Length Std	6-Flow IAT Min	11-Flow IAT Mean	16-Fwd Packet Length Mean
2-Flow Bytes/s	7-Fwd IAT Total	12-Total Length of Bwd Packets	17-Total Backward Packets
3-Total Length of Fwd Packets	8-Flow Duration	13-Fwd Packet Length Min	18-Total Fwd Packets
4-Fwd Packet Length Std	9-Bwd Packet Length Max	14-Bwd Packet Length Mean	19-Fwd Packet Length Max
5-Flow IAT Std	10-Flow IAT Max	15-Flow Packets/s	20-Bwd Packet Length Min

Figure 20. Feature List and F-Measure Changes in Different Feature Selections of QDA Algorithm

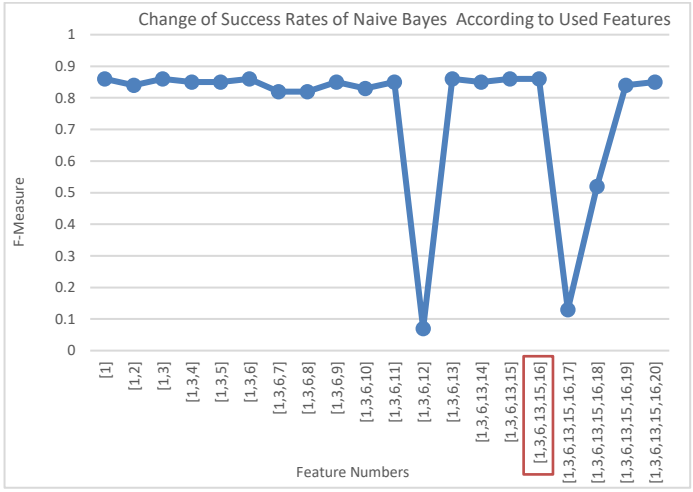
When creating this graph, the 20 features[6] are applied to the QDA algorithm in order, starting from the feature with the highest significance weight. If a decrease in the F-measure value is observed, the last inserted feature is removed from this list. At the end of this process, the feature list that provides the highest value is determined as the final feature list of this algorithm.

When the graph is examined, it is observed that very sharp decreases occur on the F-measure level, especially when the features 3,11,16 and 17 are included in the list. From these features,

[6] The features obtained in Feature Selection According to Attack or Benign section are used.

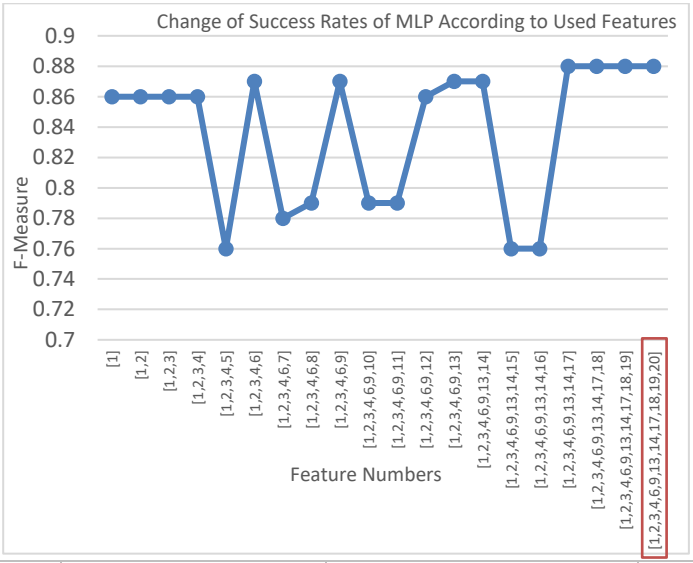
the presence of the 3rd feature (`Fwd Packet Length Std`) in both of the implementation that forms tables 8 and 9 may be the likely cause of low QDA scores. In this context, the application final result is updated by applying the feature group which obtains the highest value in the graph (0, 1, 2, and 5th features).

Moreover, this method has a significant increase in results when it applied to Naive Bayes and MLP algorithms, also. It can be seen in the Figure 21 and Figure 22 that illustrates how these two algorithms respond to the possibilities of different feature selection.



1-Bwd Packet Length Std	6-Flow IAT Min	11-Flow IAT Mean	16-Fwd Packet Length Mean
2-Flow Bytes/s	7-Fwd IAT Total	12-Total Length of Bwd Packets	17-Total Backward Packets
3-Total Length of Fwd Packets	8-Flow Duration	13-Fwd Packet Length Min	18-Total Fwd Packets
4-Fwd Packet Length Std	9-Bwd Packet Length Max	14-Bwd Packet Length Mean	19-Fwd Packet Length Max
5-Flow IAT Std	10-Flow IAT Max	15-Flow Packets/s	20-Bwd Packet Length Min

Figure 21. F-Measure Changes in Different Feature Selections of Naive Bayes Algorithm



1-Bwd Packet Length Std	6-Flow IAT Min	11-Flow IAT Mean	16-Fwd Packet Length Mean
2-Flow Bytes/s	7-Fwd IAT Total	12-Total Length of Bwd Packets	17-Total Backward Packets
3-Total Length of Fwd Packets	8-Flow Duration	13-Fwd Packet Length Min	18-Total Fwd Packets
4-Fwd Packet Length Std	9-Bwd Packet Length Max	14-Bwd Packet Length Mean	19-Fwd Packet Length Max
5-Flow IAT Std	10-Flow IAT Max	15-Flow Packets/s	20-Bwd Packet Length Min

Figure 22. F-Measure Changes in Different Feature Selections of MLP Algorithm

In this context, if the selected features are updated with the data obtained from the last three graphs according to the algorithms used, the F-Measure scores of these three algorithms (Naive Bayes, QDA and MLP) will increase. Updated features can be seen in Table 10.

Algorithms	**Features**
Naive Bayes	Bwd Packet Length Std, Total Length of Fwd Packets, Flow IAT Min, Fwd Packet Length Min, Flow Packets/s, Fwd Packet Length Mean
QDA	Bwd Packet Length Std, Flow Bytes/s, Total Length of Fwd Packets, Flow IAT Min
MLP	Bwd Packet Length Std, Flow Bytes/s, Total Length of Fwd Packets, Fwd Packet Length Std, Flow IAT Min, Bwd Packet Length Max, Fwd Packet Length Min, Bwd Packet Length Mean, Total Backward Packets, Total Fwd Packets, Fwd Packet Length Max, Bwd Packet Length Min
Random Forest	Bwd Packet Length Std, Flow Bytes/s, Total Length of Fwd Packets, Fwd Packet Length Std, Flow IAT Std, Flow IAT Min, Fwd IAT Total (No changes were made to this feature list. The features obtained in "3.2.3.2 Feature Selection According to Attack or Benign" section are used.)
ID3	
AdaBoost	
K Nearest Neighbours	

Table 10. According to Machine Learning Algorithms updated features.

If the application is updated according to the properties contained in Table 10, the following results are obtained. Changed values are highlighted in red.

Machine Learning Algorithms	**Evaluation Criteria**				
	F-Measure	Precision	Recall	Accuracy	Time
Naive Bayes	0.86	0.86	0.87	0.87	**1.8255**
QDA	0.86	0.87	0.88	0.88	2.3696
Random Forest	0.94	0.94	0.94	0.94	19.0899
ID3	0.95	0.95	0.95	0.95	9.5107
AdaBoost	0.94	0.94	0.94	0.94	135.2455
MLP	*0.83*	0.82	0.87	0.87	59.6933
K Nearest Neighbours	**0.97**	0.97	0.97	0.97	*1626.833*

Table 11. The Final Results - Implementation of features using Table 10.

When Table 11, which contains the final results of this section, is examined, according to the previous results, an increase of 0.5 points in Naive Bayes and 0.4 points in MLP is observed. However, the biggest change occurred in the QDA. It reached 0.86 with an increase of 0.45 points.

Thus, the MLP with the lowest performance of 0.83, while the K Nearest Neighbours algorithm has the highest score as in all applications. As for speed, Naive Bayes is the fastest algorithm in all applications, while KNN is the slowest.

5 Evaluation

In this section, the final results of the implementation (See Table 11) are compared with a study in the literature. As a study to be compared, the study conducted by Sharafaldin et al [4], in 2017 was chosen. The reason for this is that the mentioned article has similar steps to our work such as similar machine learning methods, feature selection, and dataset.

The F-measure was determined as the main evaluation criterion. However, in cases where equality prevailed in the results, Recall and Precision were assessed respectively. Sharafaldin and his colleagues [4] did not specify the hardware information used during the implementation phase. Therefore, the timing of the execution was not included in the evaluation, considering that it may be misleading, even though it is shared in both works.

Machine Learning Algorithms	**Our Study**			**Sharafaldin et al[4].**		
	F-Measure	Precision	Recall	F-Measure	Precision	Recall
Naive Bayes	**0.86**	0.86	0.87	0.84[7]	0.88	0.84[7]
QDA	0.86	0.87	0.88	**0.92**	0.97	0.88
Random Forest	0.94	0.94	0.94	**0.97**	0.98	0.97
ID3	0.95	0.95	0.95	**0.98**	0.98	0.98
AdaBoost	**0.94**	0.94	0.94	0.77	0.77	0.84
MLP	**0.83**	0.82	0.87	0.76	0.77	0.83
K Nearest Neighbours	**0.97**	0.97	0.97	0.96	0.96	0.96

Table 12. Comparison of the performance of two exercises against the evaluation criteria.

Table 12 and Figure 23 show the comparison of the results obtained from two studies. When the results obtained in them are examined, it is seen that Random Forest, ID3 and QDA algorithms of Sharafaldin and his colleagues [4] are 0.3,0.3,0.6 points higher than our study, respectively. This big difference in QDA is quite remarkable. Because, in our study, this algorithm has a maximum of 0.86 as a result of attempts to increase performance

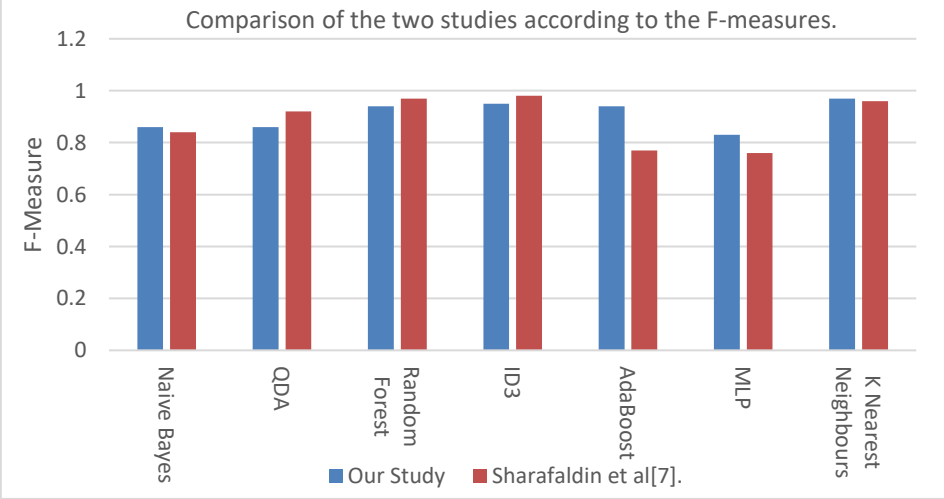


Figure 23. Comparison of the performance of the two studies compared to the F-measures.

[7] In the published article [7], this value appears as "0.04". However, by communicating with the author, it has been learned that there is a typing error. So, the value has been updated.

On the other hand, the F-criterion score of our study is 0.1 and 0.2 points higher in the KNN and Naive Bayes algorithms, respectively. In the Adaboost and MLP methods, the difference is remarkably high (0.17 and 0.7) In both studies, MLP is the method with the lowest success rate. However, it is possible to increase performance by increasing the hidden layer size and depth in MLP. In our study, an MLP with a 3-step hidden layer and 13 neurons in each layer was used, but in the other study, the comparison is not possible since no information is given on this topic.

The highest performance was in the KNN algorithm with 0.97 points in our study whereas in the other study ID3 with 0.98 points. When speed is taken into account, in both studies Naive Bayes is the fastest algorithm, while the KNN is the slowest.

6 Conclusion and Future Work

6.1 Conclusion

In this study, it is aimed to detect network anomaly using machine learning methods. In this context, the CICIDS2017[16] has been used as dataset because of its up-to-datedness, wide attack diversity, and various network protocols (e.g. Mail services, SSH, FTP, HTTP, and HTTPS)[4]. This dataset contains more than 80 features that define the network flow. During the application, the importance weight calculation was made with the Random Forest Regressor[63] algorithm to decide which of these features will be used in machine learning methods. Two approaches have been used when making these calculations. In the first place, importance weights are calculated separately for each attack type. In the second method, all the attacks are collected under a single group and the importance weights for this group are calculated. that is, the common properties that are important for all attacks are determined. Finally, seven machine learning algorithms, which are widely used and have different qualities, have been applied to this data. These algorithms and the achieved performance ratios according to F-measure are as follows (F-measure takes a value between 0 and 1): Naive Bayes: 0.86, QDA: 0.86, Random Forest: 0.94, ID3: 0.95, AdaBoost: 0.94, MLP: 0.83, and K Nearest Neighbours: 0.97.

6.2 Future Work

This work is open to future improvements. In this section, a few of these improvement possibilities will be presented.

In this study, a data set consisting of CSV files containing features obtained from the network flow was used as the training and test data. Unfortunately, this method is not practically viable in real systems. However, this problem can be solved by adding a module that catches real network data and makes it workable with the machine learning algorithm.

Another point was that during this study, various machine learning methods were applied independently from each other and experimental results were obtained. However, this method has a weak practical applicability in real life. In order to deal with this problem, a multi-layered / hierarchical machine learning structure can be designed. Moreover, thanks to such a structure, it is possible to save time, CPU power and memory. For example, in a two-tiered structure, the

first layer can be created from fast and computationally cheap algorithms such as Naive Bayes or QDA, so network traffic can be observed continuously and at minimal cost. The first step, when detecting any anomaly, transmits it to an upper layer composed of algorithms with higher performance level. This layer measures by forming the decision mechanism. The first step, when detecting an anomaly, transmits it to an upper layer composed of algorithms with higher performance such as ID3, AdaBoost, and KNN. The final layer that makes up the determination mechanism takes the precautionary decision to protect the network against attack.

References

[1] K. Kostas, "Anomaly Detection in Networks Using Machine Learning," Research Proposal, 23 Mar 2018, 2018.

[2] “Internet Growth Statistics,” *Miniwatts Marketing Group, 2 Mar 2018.* [Online]. Available: https://www.internetworldstats.com/emarketing.htm. [Accessed 26 Aug 2018].

[3] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, 2005, pp. 333-342: Australian Computer Society, Inc.

[4] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Software Networking*, vol. 1, no. 1, pp. 177-200, 2017.

[5] "1998 DARPA Intrusion Detection Evaluation Data Set," *Lincoln Laboratory, Massachusetts Institute of Technology,* [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-data-set. [Accessed 05 Aug 2018].

[6] C. Thomas, V. Sharma, and N. Balakrishnan, "Usefulness of DARPA dataset for intrusion detection system evaluation," in *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008*, 2008, vol. 6973, p. 69730G: International Society for Optics and Photonics.

[7] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *Information Science and Security (ICISS), 2016 International Conference on*, 2016, pp. 1-6: IEEE.

[8] "KDD Cup 1999 Data," *University of California, Irvine,* [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Accessed 05 Aug 2018].

[9] A. Özgür and H. Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ PrePrints*, vol. 4, p. e1954v1, 2016.

[10] "CAIDA OC48 Peering Point Traces dataset," *Center for Applied Internet Data Analysis*, [Online]. Available: https://www.caida.org/data/passive/passive_oc48_dataset.xml. [Accessed06 Aug 2018].

[11] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19-31, 2016.

[12] "NSL-KDD dataset," *Canadian Institute for Cybersecurity, University of New Brunswick,* [Online]. Available: http://www.unb.ca/cic/datasets/nsl.html. [Accessed 06 Aug 2018].

[13] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, 2009, pp. 1-6: IEEE.

[14] "Intrusion detection evaluation dataset (ISCXIDS2012)," *Canadian Institute for Cybersecurity, University of New Brunswick,* [Online]. Available: http://www.unb.ca/cic/datasets/ids.html. [Accessed 07 Aug 2018].

[15] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357-374, 2012.

[16] "Intrusion Detection Evaluation Dataset (CICIDS2017)," *Canadian Institute for Cybersecurity, University of New Brunswick,* [Online]. Available: http://www.unb.ca/cic/datasets/ids-2017.html. [Accessed 08 Aug 2018].
[17] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer security: principles and practice*. Pearson Education, 2012.
[18] M. Chhabra, B. Gupta, and A. Almomani, "A novel solution to handle DDOS attack in MANET," *Journal of Information Security,* vol. 4, no. 03, p. 165, 2013.
[19] "Hulk DoS tool," *GitHub Inc,* [Online]. Available: https://github.com/grafov/hulk. [Accessed 08 Aug 2018].
[20] S. Behal and K. Kumar, "Characterization and Comparison of DDoS Attack Tools and Traffic Generators: A Review," *IJ Network Security,* vol. 19, no. 3, pp. 383-393, 2017.
[21] S. Haris, R. Ahmad, and M. Ghani, "Detecting TCP SYN flood attack based on anomaly detection," in *Network Applications Protocols and Services (NETAPPS), 2010 Second International Conference on*, 2010, pp. 240-244: IEEE.
[22] J. Choi, C. Choi, B. Ko, D. Choi, and P. Kim, "Detecting Web based DDoS Attack using MapReduce operations in Cloud Computing Environment," *J. Internet Serv. Inf. Secur.*, vol. 3, no. 3/4, pp. 28-37, 2013.
[23] "Nmap: *the Network Mapper - Free Security Scanner," Nmap.org. (2018),* [online] Available: https://nmap.org/ [Accessed 10 Aug. 2018].
[24] R. Christopher, "Port scanning techniques and the defense against them," *SANS Institute,* 2001.
[25] J. Gadge and A. A. Patil, "Port scan detection," in *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, 2008, pp. 1-6: IEEE.
[26] "LOIC," *SourceForge,* [Online]. Available: https://sourceforge.net/projects/loic/. [Accessed: 11-Aug-2018].
[27] "GoldenEye," *GitHub, 20-Jun-2018.* [Online]. Available: Available: https://github.com/jseidl/GoldenEye. [Accessed: 11-Aug-2018].
[28] "Patator," *GitHub, 04-Aug-2018.* [Online]. Available: https://github.com/lanjelot/patator. [Accessed: 11-Aug-2018].
[29] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, and R. Zuech, "Machine learning for detecting brute force attacks at the network level," in *Bioinformatics and Bioengineering (BIBE), 2014 IEEE International Conference on*, 2014, pp. 379-385: IEEE.
[30] T. Ylonen and C. Lonvick, "The secure shell (SSH) protocol architecture," 2070-1721, 2005.
[31] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: a flow-based SSH intrusion detection system," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, 2012, pp. 86-97: Springer.
[32] "A new DOS Perl Program," *GitHub, 05 Nov 2013.* [Online]. Available: https://github.com/llaera/slowloris.pl. [Accessed: 12 Aug 2018].
[33] "slowhttptest," *GitHub.* [Online]. Available: https://github.com/shekyan/slowhttptest/wiki. [Accessed: 12 Aug 2018].
[34] "Slow Read DoS Attack." *Istanbul Technical University, 07 Sep 2013.* [Online]. Available: https://bidb.itu.edu.tr/eskiler/seyirdefteri/blog/2013/09/07/slow-read-dos-attack-(yavaş-okutarak-hizmet-engelleme-saldırısı). [Accessed: 12 Aug 2018].
[35] "Ares," *GitHub, 08-Dec-2017.* [Online]. Available: https://github.com/sweetsoftware/Ares. [Accessed: 12 Aug 2018].
[36] A. Sperotto, *Flow-based intrusion detection*. Citeseer, 2010.

[37] P. Likarish, E. Jung, and I. Jo, "Obfuscated malicious javascript detection using classification techniques," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, 2009, pp. 47-54: IEEE.
[38] "heartleech," *GitHub, 07-Jun-2014.* [Online]. Available: https://github.com/robertdavidgraham/heartleech. [Accessed: 13 Aug 2018].
[39] V. Bartoš, "Heartbleed Detection at CESNET using Extended Flow Monitoring," in *Proceedings of 8th International Scientific Conference on Security and Protection of Information*, 2015.
[40] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2017.
[41] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3-24, 2007.
[42] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5-32, 2001.
[43] *J. P. Mueller and L. Massaron, "Machine Learning For Dummies Cheat Sheet," dummies. [Online]*. Available: https://www.dummies.com/programming/big-data/data-science/machine-learning-dummies-cheat-sheet/. [Accessed: 14 Aug 2018].
[44] R. E. Schapire, "The boosting approach to machine learning: An overview," in *Nonlinear estimation and classification*: Springer, 2003, pp. 149-171.
[45] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.
[46] M. S. Uzer, "Feature Selection Algorithms Developed by Using Artificial Intelligence And Feature Transform Methods In Pattern Recognition Applications," Ph.D Thesis, The Graduate School of Natural and Applied Science Selçuk University, 2014.
[47] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & security*, vol. 24, no. 4, pp. 295-307, 2005.
[48] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB journal*, vol. 16, no. 4, pp. 507-521, 2007.
[49] M. Suresh and R. Anitha, "Evaluating machine learning algorithms for detecting DDoS attacks," in *International Conference on Network Security and Applications*, 2011, pp. 441-452: Springer.
[50] S. Mukherjee and N. Sharma, "Intrusion detection using naive Bayes classifier with feature reduction," *Procedia Technology*, vol. 4, pp. 119-128, 2012.
[51] W. Yassin, N. I. Udzir, Z. Muda, and M. N. Sulaiman, "Anomaly-based intrusion detection through k-means clustering and naives bayes classification," in *Proc. 4th Int. Conf. Comput. Informatics, ICOCI*, 2013, no. 49, pp. 298-303.
[52] "Python 3.6.6 documentation," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3.6/. [Accessed: 18-Aug-2018].
[53] *"scikit-learn," scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 18-Aug-2018].
[54] "Python Data Analysis Library," *pandas: powerful Python data analysis toolkit - pandas 0.22.0 documentation.* [Online]. Available: https://pandas.pydata.org/. [Accessed: 18-Aug-2018].
[55] "Matplotlib: *Python plotting - Matplotlib 2.2.2 documentation," Matplotlib.* [Online]. Available: https://matplotlib.org/. [Accessed: 18-Aug-2018].
[56] "NumPy*," NumPy developers.* [Online]. Available: http://www.numpy.org/. [Accessed: 18-Aug-2018].

[57] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Ieee communications surveys & tutorials,* vol. 16, no. 1, pp. 303-336, 2014.

[58] "sklearn.preprocessing.LabelEncoder," *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. [Accessed: 19-Aug-2018].

[59] "train_test_split," *scikit-learn 0.19.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. [Accessed: 20-Aug-2018].

[60] *J. Brownlee, "Feature Importance and Feature Selection With XGBoost in Python," Machine Learning Mastery, 10-Mar-2018. [Online]*. Available: https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/. [Accessed: 19-Aug-2018].

[61] R. Alshammari and A. N. Zincir-Heywood, "A flow based approach for SSH traffic detection," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, 2007, pp. 296-301: IEEE.

[62] H. Choi, H. Lee, and H. Kim, "Fast detection and visualization of network attacks on parallel coordinates," *computers & security,* vol. 28, no. 5, pp. 276-288, 2009.

[63] "sklearn.ensemble.RandomForestRegressor," *scikit-learn.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed: 19-Aug-2018].

Appendix A. The List of The Features and Explanations

This list has been taken from: http://www.netflowmeter.ca/netflowmeter.html

№	Feature Name	Feature Description
1	Flow ID	Flow ID
2	Source IP	Source IP
3	Source Port	Source Port
4	Destination IP	Destination IP
5	Destination Port	Destination Port
6	Protocol	Protocol
7	Timestamp	Timestamp
8	Flow Duration	Duration of the flow in Microsecond
9	Total Fwd Packets	Total packets in the forward direction
10	Total Backward Packets	Total packets in the backward direction
11	Total Length of Fwd Packets	Total size of packet in forward direction
12	Total Length of Bwd Packets	Total size of packet in backward direction
13	Fwd Packet Length Max	Maximum size of packet in forward direction
14	Fwd Packet Length Min	Minimum size of packet in forward direction
15	Fwd Packet Length Mean	Mean size of packet in forward direction
16	Fwd Packet Length Std	Standard deviation size of packet in forward direction
17	Bwd Packet Length Max	Maximum size of packet in backward direction
18	Bwd Packet Length Min	Minimum size of packet in backward direction
19	Bwd Packet Length Mean	Mean size of packet in backward direction
20	Bwd Packet Length Std	Standard deviation size of packet in backward direction
21	Flow Bytes/s	Number of flow bytes per second
22	Flow Packets/s	Number of flow packets per second
23	Flow IAT Mean	Mean length of a flow
24	Flow IAT Std	Standard deviation length of a flow
25	Flow IAT Max	Maximum length of a flow
26	Flow IAT Min	Minimum length of a flow
27	Fwd IAT Total	Total time between two packets sent in the forward direction
28	Fwd IAT Mean	Mean time between two packets sent in the forward direction
29	Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
30	Fwd IAT Max	Maximum time between two packets sent in the forward direction
31	Fwd IAT Min	Minimum time between two packets sent in the forward direction
32	Bwd IAT Total	Total time between two packets sent in the backward direction
33	Bwd IAT Mean	Mean time between two packets sent in the backward direction
34	Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
35	Bwd IAT Max	Maximum time between two packets sent in the backward direction
36	Bwd IAT Min	Minimum time between two packets sent in the backward direction
37	Fwd PSH Flags	Number of packets with PUSH
38	Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
39	Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
40	Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
41	Fwd Header Length	Total bytes used for headers in the forward direction
42	Bwd Header Length	Total bytes used for headers in the backrward direction
43	Fwd Packets/s	Number of forward packets per second
44	Bwd Packets/s	Number of backward packets per second

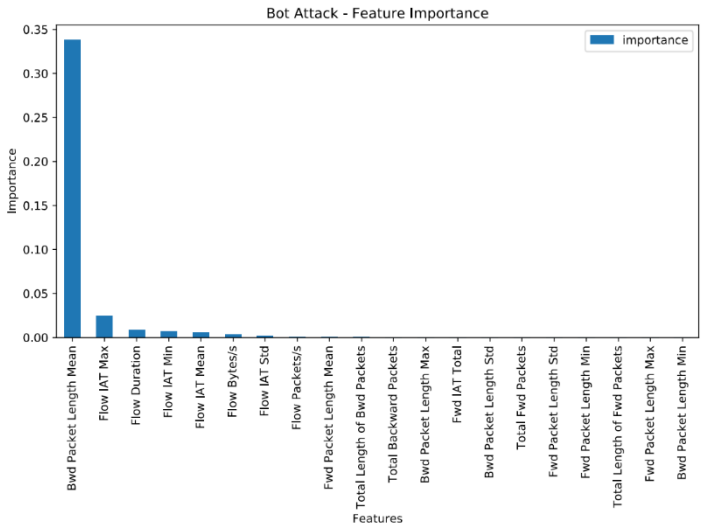
Appendix A Figure 1. This list includes the features and explanations contained in the CICIDS2017 dataset.

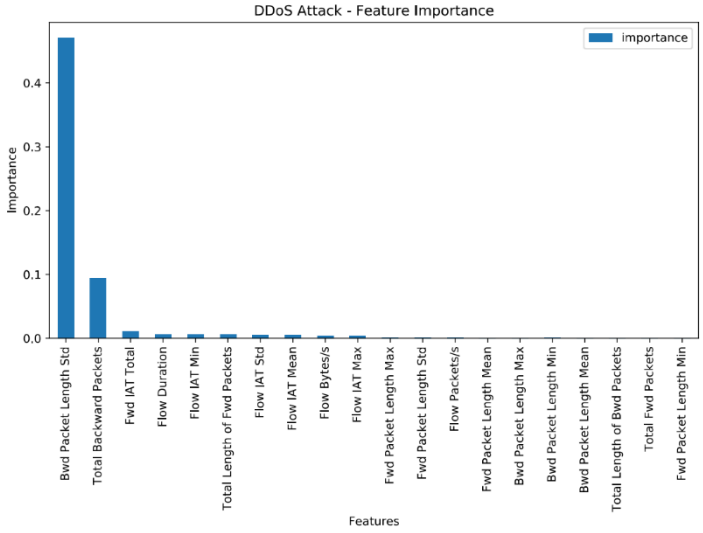
This list has been taken from: http://www.netflowmeter.ca/netflowmeter.html

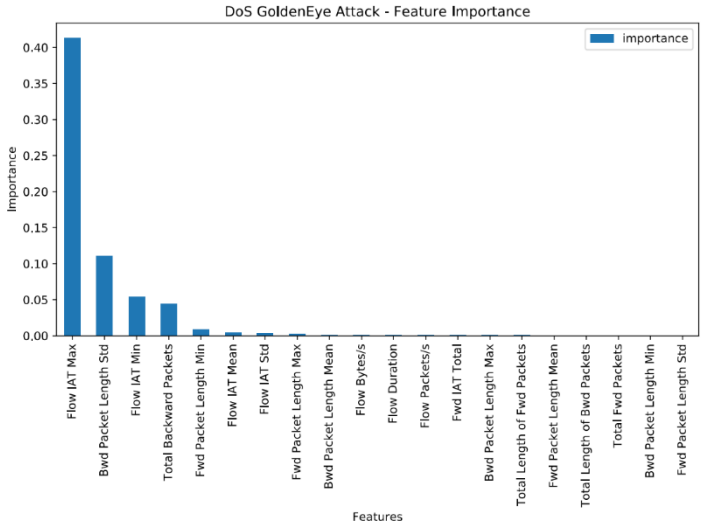
№	Feature Name	Feature Description
45	Min Packet Length	Minimum inter-arrival time of packet
46	Max Packet Length	Maximum inter-arrival time of packet
47	Packet Length Mean	Mean inter-arrival time of packet
48	Packet Length Std	Standard deviation inter-arrival time of packet
49	Packet Length Variance	Packet Length Variance
50	FIN Flag Count	Number of packets with FIN
51	SYN Flag Count	Number of packets with SYN
52	RST Flag Count	Number of packets with RST
53	PSH Flag Count	Number of packets with PUSH
54	ACK Flag Count	Number of packets with ACK
55	URG Flag Count	Number of packets with URG
56	CWE Flag Count	Number of packets with CWE
57	ECE Flag Count	Number of packets with ECE
58	Down/Up Ratio	Download and upload ratio
59	Average Packet Size	Average size of packet
60	Avg Fwd Segment Size	Average size observed in the forward direction
61	Avg Bwd Segment Size	Average size observed in the backward direction
62	Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
63	Fwd Avg Packets/Bulk	Average number of packets bulk rate in the forward direction
64	Bwd Avg Bulk Rate	Average number of bulk rate in the backward direction
65	Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
66	Bwd Avg Packets/Bulk	Average number of packets bulk rate in the backward direction
67	Bwd Avg Bulk Rate	Average number of bulk rate in the backward direction
68	Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
69	Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
70	Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
71	Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
72	Init_Win_bytes_forward	The total number of bytes sent in initial window in the forward direction
73	Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
74	act_data_pkt_fwd	Count of packets with at least 1 byte of TCP data payload in the forward direction
75	min_seg_size_forward	Minimum segment size observed in the forward direction
76	Active Mean	Mean time a flow was active before becoming idle
77	Active Std	Standard deviation time a flow was active before becoming idle
78	Active Max	Maximum time a flow was active before becoming idle
79	Active Min	Minimum time a flow was active before becoming idle
80	Idle Mean	Mean time a flow was idle before becoming active
81	Idle Std	Standard deviation time a flow was idle before becoming active
82	Idle Max	Maximum time a flow was idle before becoming active
83	Idle Min	Minimum time a flow was idle before becoming active
84	Label	Label
85	External IP	External IP

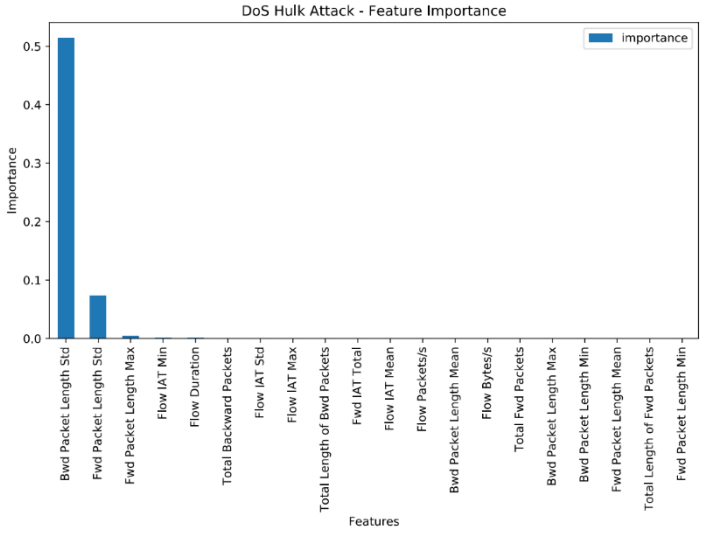
Appendix A Figure 2. This list includes the features and explanations contained in the CICIDS2017 dataset.

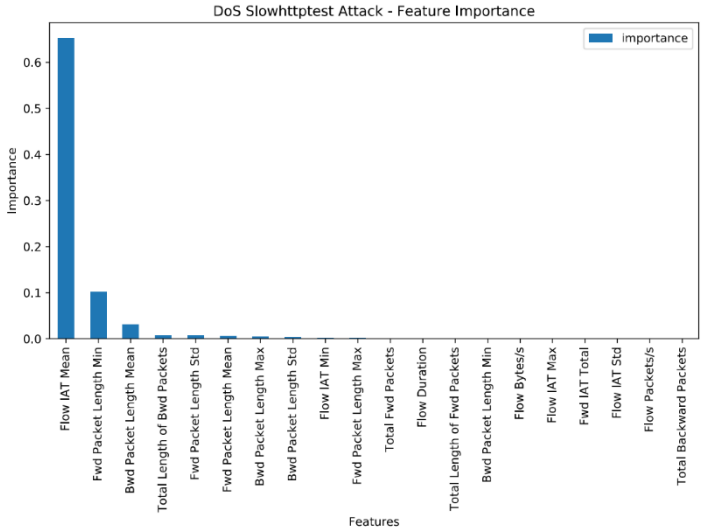
Appendix B the Importance Weights of Features, According to Attacks

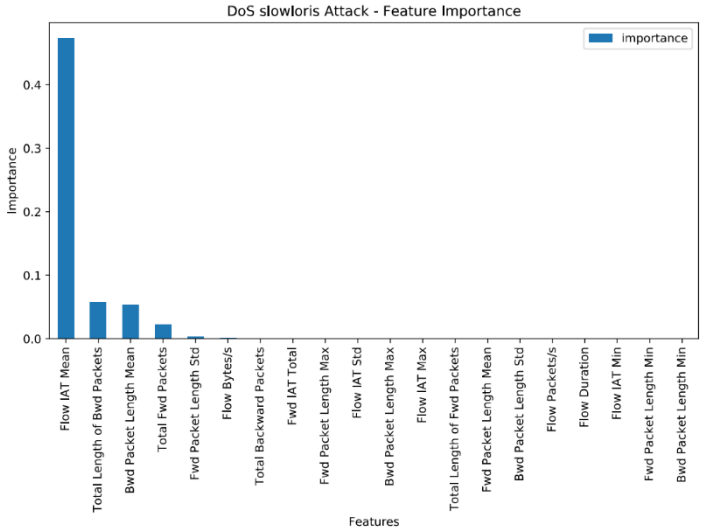


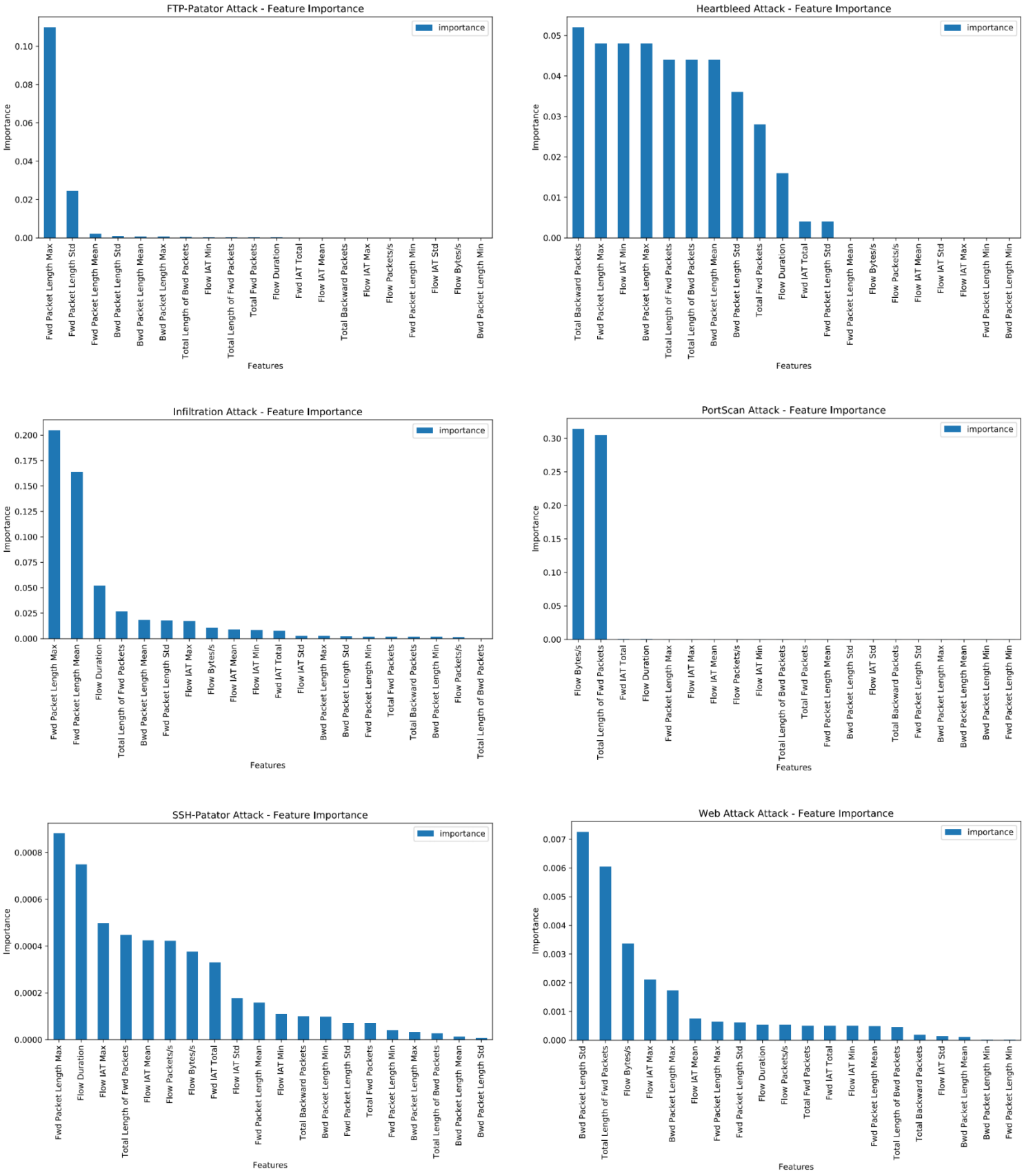












Appendix B Figure 2. The importance Weights Graphics of Features According to Attacks

Bot attack importance list:

	Features	importance
1	Bwd Packet Length Mean	0.338129
2	Flow IAT Max	0.024407
3	Flow Duration	0.008495
4	Flow IAT Min	0.007067
5	Flow IAT Mean	0.005609
6	Flow Bytes/s	0.003518
7	Flow IAT Std	0.002213
8	Flow Packets/s	0.000667
9	Fwd Packet Length Mean	0.000418
10	Total Length of Bwd Packets	0.000333
11	Total Backward Packets	0.000223
12	Bwd Packet Length Max	0.000212
13	Fwd IAT Total	0.000141
14	Bwd Packet Length Std	0.000083
15	Total Fwd Packets	0.000077
16	Fwd Packet Length Std	0.000075
17	Fwd Packet Length Min	0.000069
18	Total Length of Fwd Packets	0.000048
19	Fwd Packet Length Max	0.000036
20	Bwd Packet Length Min	0.000005

DoS Slowhttptest attack importance list:

	Features	importance
1	Flow IAT Mean	0.653023
2	Fwd Packet Length Min	0.101739
3	Bwd Packet Length Mean	0.029976
4	Total Length of Bwd Packets	0.007153
5	Fwd Packet Length Std	0.007107
6	Fwd Packet Length Mean	0.006074
7	Bwd Packet Length Max	0.003857
8	Bwd Packet Length Std	0.002934
9	Flow IAT Min	0.002184
10	Fwd Packet Length Max	0.001245
11	Total Fwd Packets	0.000808
12	Flow Duration	0.000802
13	Total Length of Fwd Packets	0.000604
14	Bwd Packet Length Min	0.000545
15	Flow Bytes/s	0.000421
16	Flow IAT Max	0.000327
17	Fwd IAT Total	0.000284
18	Flow IAT Std	0.000252
19	Flow Packets/s	0.000159
20	Total Backward Packets	0.000130

Infiltration attack importance list:

	Features	importance
1	Fwd Packet Length Max	0.204751
2	Fwd Packet Length Mean	0.163696
3	Flow Duration	0.052250
4	Total Length of Fwd Packets	0.026823
5	Bwd Packet Length Mean	0.018539
6	Fwd Packet Length Std	0.017846
7	Flow IAT Max	0.017182
8	Flow Bytes/s	0.010812
9	Flow IAT Mean	0.008816
10	Flow IAT Min	0.008310
11	Fwd IAT Total	0.007726
12	Flow IAT Std	0.002862
13	Bwd Packet Length Max	0.002737
14	Bwd Packet Length Std	0.002030
15	Fwd Packet Length Min	0.001937
16	Total Fwd Packets	0.001862
17	Total Backward Packets	0.001680
18	Bwd Packet Length Min	0.001610
19	Flow Packets/s	0.001406
20	Total Length of Bwd Packets	0.000000

DDoS attack importance list:

	Features	importance
1	Bwd Packet Length Std	0.471368
2	Total Backward Packets	0.093576
3	Fwd IAT Total	0.010827
4	Flow Duration	0.006320
5	Flow IAT Min	0.006110
6	Total Length of Fwd Packets	0.006037
7	Flow IAT Std	0.005439
8	Flow IAT Mean	0.005238
9	Flow Bytes/s	0.004741
10	Flow IAT Max	0.004703
11	Fwd Packet Length Max	0.001647
12	Fwd Packet Length Std	0.001406
13	Flow Packets/s	0.001019
14	Fwd Packet Length Mean	0.000619
15	Bwd Packet Length Max	0.000577
16	Bwd Packet Length Min	0.000405
17	Bwd Packet Length Mean	0.000197
18	Total Length of Bwd Packets	0.000085
19	Total Fwd Packets	0.000042
20	Fwd Packet Length Min	0.000017

DoS slowloris attack importance list:

	Features	importance
1	Flow IAT Mean	0.473856
2	Total Length of Bwd Packets	0.057384
3	Bwd Packet Length Mean	0.053022
4	Total Fwd Packets	0.021895
5	Fwd Packet Length Std	0.002831
6	Flow Bytes/s	0.000965
7	Total Backward Packets	0.000769
8	Fwd IAT Total	0.000755
9	Fwd Packet Length Max	0.000728
10	Flow IAT Std	0.000664
11	Bwd Packet Length Max	0.000608
12	Flow IAT Max	0.000603
13	Total Length of Fwd Packets	0.000579
14	Fwd Packet Length Mean	0.000537
15	Bwd Packet Length Std	0.000528
16	Flow Packets/s	0.000429
17	Flow Duration	0.000375
18	Flow IAT Min	0.000348
19	Fwd Packet Length Min	0.000152
20	Bwd Packet Length Min	0.000021

PortScan attack importance list:

	Features	importance
1	Flow Bytes/s	0.313933
2	Total Length of Fwd Packets	0.304613
3	Fwd IAT Total	0.000427
4	Flow Duration	0.000398
5	Fwd Packet Length Max	0.000150
6	Flow IAT Max	0.000059
7	Flow IAT Mean	0.000054
8	Flow Packets/s	0.000031
9	Flow IAT Min	0.000031
10	Total Length of Bwd Packets	0.000022
11	Total Fwd Packets	0.000021
12	Fwd Packet Length Mean	0.000019
13	Bwd Packet Length Std	0.000018
14	Flow IAT Std	0.000017
15	Total Backward Packets	0.000014
16	Fwd Packet Length Std	0.000009
17	Bwd Packet Length Max	0.000004
18	Bwd Packet Length Mean	0.000002
19	Bwd Packet Length Min	0.000002
20	Fwd Packet Length Min	0.000001

DoS GoldenEye attack importance list:

	Features	importance
1	Flow IAT Max	0.413073
2	Bwd Packet Length Std	0.111039
3	Flow IAT Min	0.054021
4	Total Backward Packets	0.044895
5	Fwd Packet Length Min	0.009531
6	Flow IAT Mean	0.004040
7	Flow IAT Std	0.003863
8	Fwd Packet Length Max	0.002899
9	Bwd Packet Length Mean	0.001228
10	Flow Bytes/s	0.000928
11	Flow Duration	0.000927
12	Flow Packets/s	0.000892
13	Fwd IAT Total	0.000719
14	Bwd Packet Length Max	0.000445
15	Total Length of Fwd Packets	0.000406
16	Fwd Packet Length Mean	0.000393
17	Total Length of Bwd Packets	0.000217
18	Total Fwd Packets	0.000065
19	Bwd Packet Length Min	0.000033
20	Fwd Packet Length Std	0.000027

FTP-Patator attack importance list:

	Features	importance
1	Fwd Packet Length Max	1.098307e-01
2	Fwd Packet Length Std	2.437956e-02
3	Fwd Packet Length Mean	2.200624e-03
4	Bwd Packet Length Std	8.997715e-04
5	Bwd Packet Length Mean	7.081365e-04
6	Bwd Packet Length Max	6.054917e-04
7	Total Length of Bwd Packets	4.015506e-04
8	Flow IAT Min	2.611238e-04
9	Total Length of Fwd Packets	1.766284e-04
10	Total Fwd Packets	1.634693e-04
11	Flow Duration	1.330083e-04
12	Fwd IAT Total	9.598814e-05
13	Flow IAT Mean	9.187611e-05
14	Total Backward Packets	8.884482e-05
15	Flow IAT Max	8.359459e-05
16	Flow Packets/s	7.030109e-05
17	Fwd Packet Length Min	4.619749e-05
18	Flow IAT Std	2.987659e-05
19	Flow Bytes/s	2.244798e-05
20	Bwd Packet Length Min	4.764849e-07

SSH-Patator attack importance list:

	Features	importance
1	Fwd Packet Length Max	0.000881
2	Flow Duration	0.000748
3	Flow IAT Max	0.000497
4	Total Length of Fwd Packets	0.000448
5	Flow IAT Mean	0.000425
6	Flow Packets/s	0.000423
7	Flow Bytes/s	0.000375
8	Fwd IAT Total	0.000329
9	Flow IAT Std	0.000177
10	Fwd Packet Length Mean	0.000158
11	Flow IAT Min	0.000111
12	Total Backward Packets	0.000100
13	Bwd Packet Length Min	0.000099
14	Fwd Packet Length Std	0.000070
15	Total Fwd Packets	0.000070
16	Fwd Packet Length Min	0.000040
17	Bwd Packet Length Max	0.000032
18	Total Length of Bwd Packets	0.000027
19	Bwd Packet Length Mean	0.000014
20	Bwd Packet Length Std	0.000008

DoS Hulk attack importance list:

	Features	importance
1	Bwd Packet Length Std	5.148222e-01
2	Fwd Packet Length Std	7.321079e-02
3	Fwd Packet Length Max	4.515548e-03
4	Flow IAT Min	1.675778e-03
5	Flow Duration	1.218072e-03
6	Total Backward Packets	3.813481e-04
7	Flow IAT Std	2.572354e-04
8	Flow IAT Max	2.517998e-04
9	Total Length of Bwd Packets	1.778769e-04
10	Fwd IAT Total	1.739909e-04
11	Flow IAT Mean	9.875828e-05
12	Flow Packets/s	8.114421e-05
13	Bwd Packet Length Mean	5.449508e-05
14	Flow Bytes/s	2.752602e-05
15	Total Fwd Packets	1.227050e-05
16	Bwd Packet Length Max	1.004453e-05
17	Bwd Packet Length Min	9.303096e-06
18	Fwd Packet Length Mean	8.013636e-06
19	Total Length of Fwd Packets	4.604820e-06
20	Fwd Packet Length Min	1.810830e-08

Heartbleed attack importance list:

	Features	importance
1	Total Backward Packets	0.052
2	Fwd Packet Length Max	0.048
3	Flow IAT Min	0.048
4	Bwd Packet Length Max	0.048
5	Total Length of Fwd Packets	0.044
6	Total Length of Bwd Packets	0.044
7	Bwd Packet Length Mean	0.044
8	Bwd Packet Length Std	0.036
9	Total Fwd Packets	0.028
10	Flow Duration	0.016
11	Fwd IAT Total	0.004
12	Fwd Packet Length Std	0.004
13	Fwd Packet Length Mean	0.000
14	Flow Bytes/s	0.000
15	Flow Packets/s	0.000
16	Flow IAT Mean	0.000
17	Flow IAT Std	0.000
18	Flow IAT Max	0.000
19	Fwd Packet Length Min	0.000
20	Bwd Packet Length Min	0.000

Web Attack attack importance list:

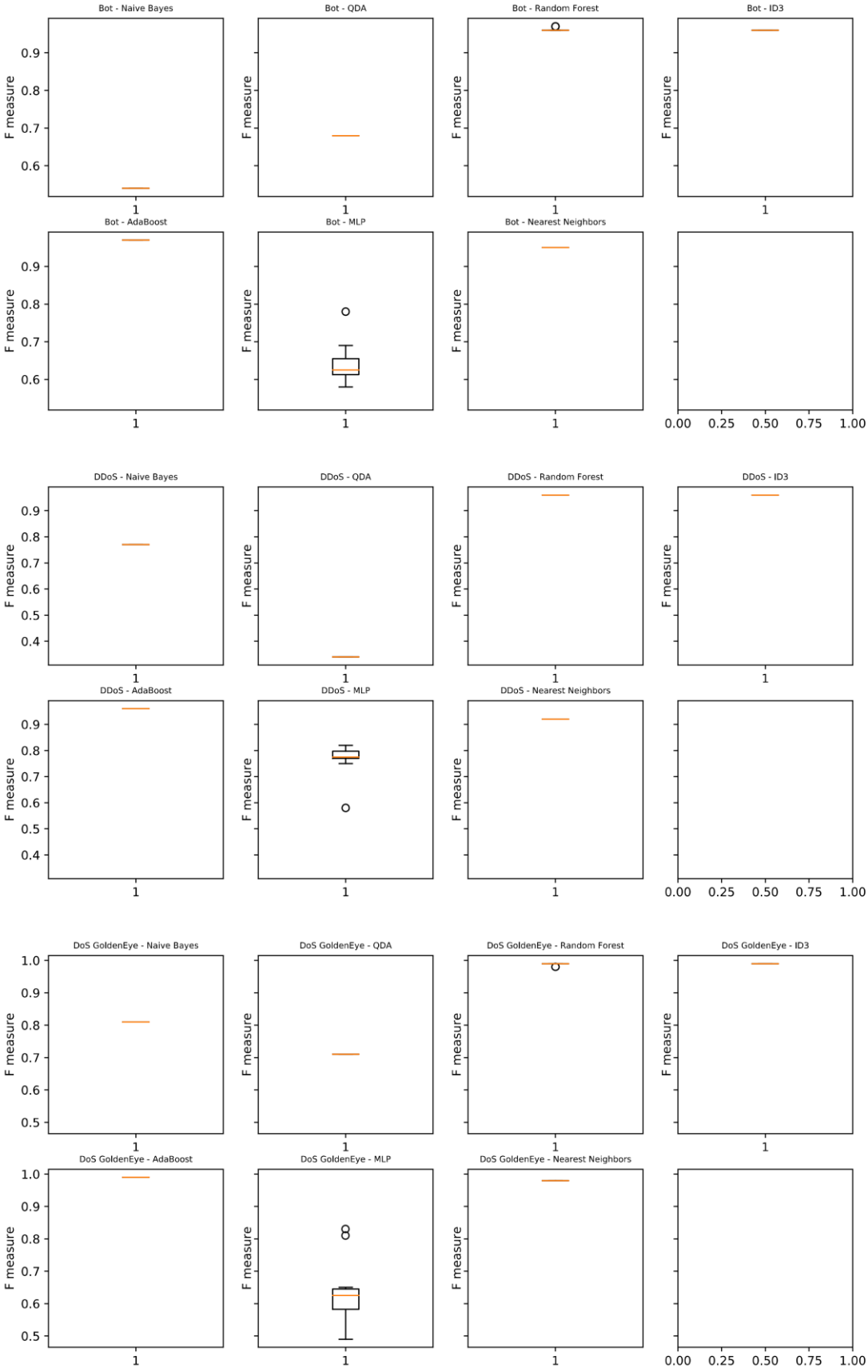
	Features	importance
1	Bwd Packet Length Std	0.007255
2	Total Length of Fwd Packets	0.006046
3	Flow Bytes/s	0.003366
4	Flow IAT Max	0.002102
5	Bwd Packet Length Max	0.001728
6	Flow IAT Mean	0.000760
7	Fwd Packet Length Max	0.000638
8	Fwd Packet Length Std	0.000616
9	Flow Duration	0.000541
10	Flow Packets/s	0.000526
11	Total Fwd Packets	0.000506
12	Fwd IAT Total	0.000505
13	Flow IAT Min	0.000499
14	Fwd Packet Length Mean	0.000490
15	Total Length of Bwd Packets	0.000454
16	Total Backward Packets	0.000177
17	Flow IAT Std	0.000140
18	Bwd Packet Length Mean	0.000102
19	Bwd Packet Length Min	0.000016
20	Fwd Packet Length Min	0.000008

Appendix B Figure 3. The importance Weights Values of Features According to Attacks.

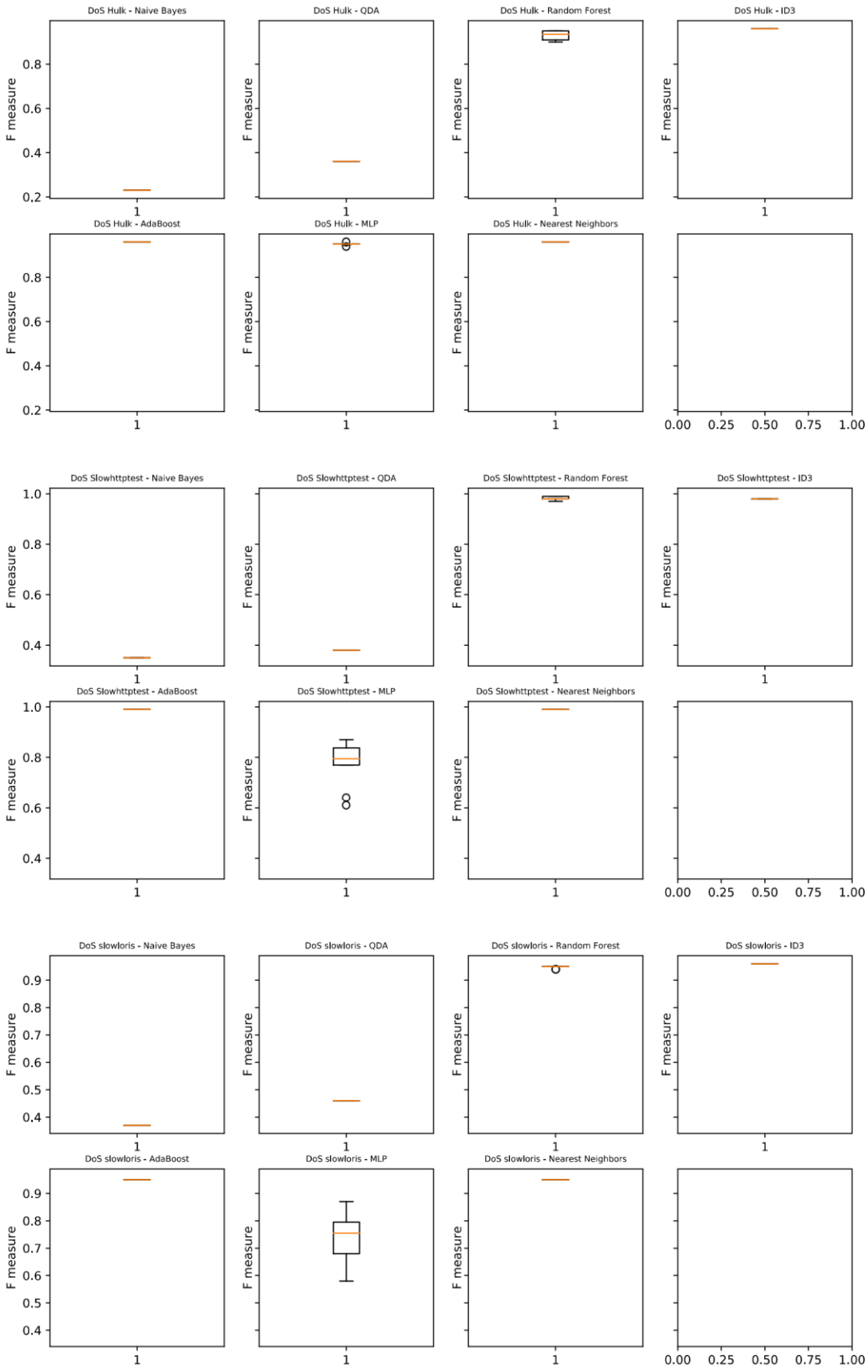
Appendix C. The Machine Learning Implementation Results (According to Attacks)

Appendix C Figure 1. Machine Learning Implementation Results (for Attack Files)

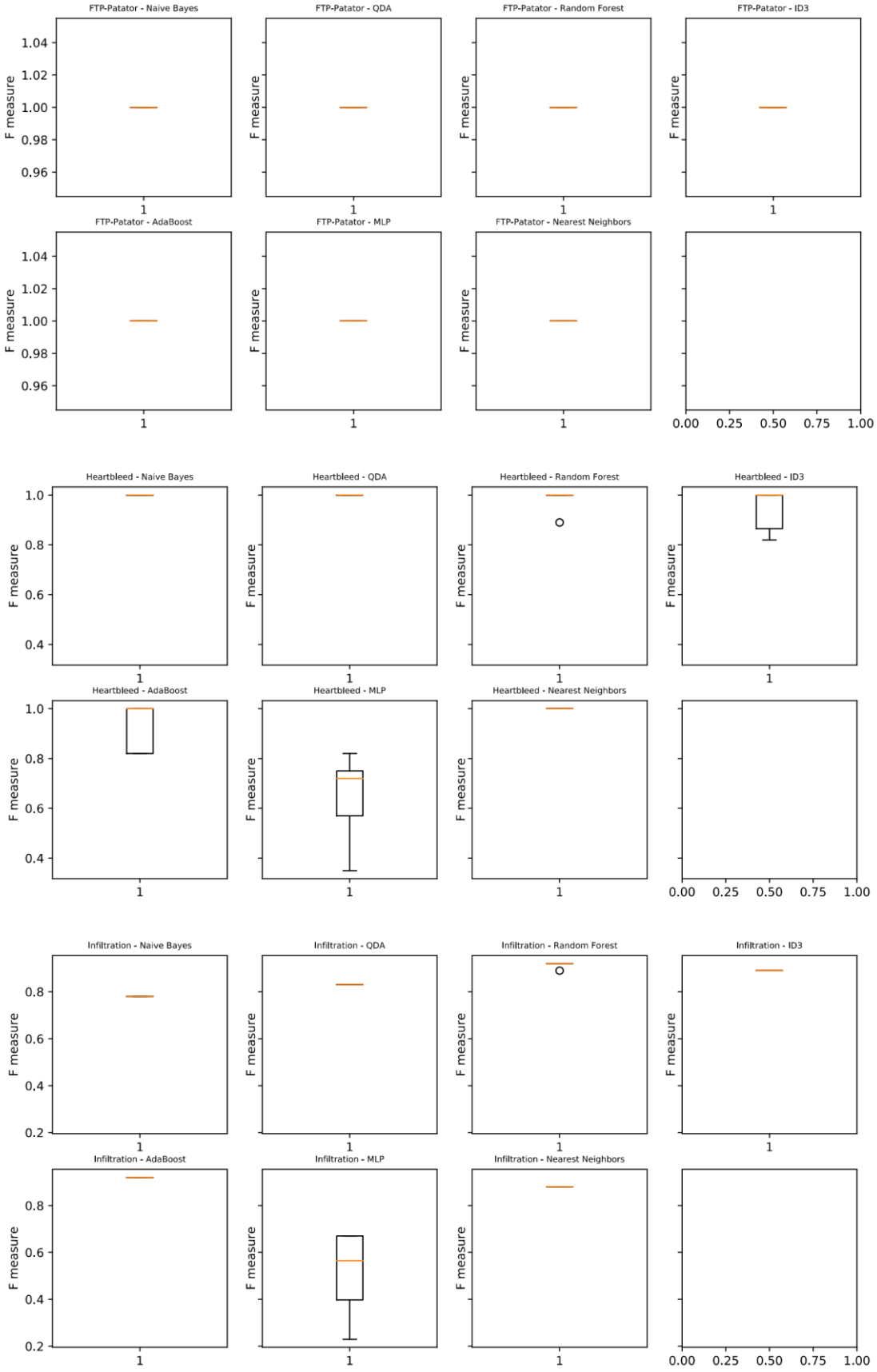
	NaiveBayes					RandomForrest					KNN					ID3					Adaboost					MLP					QDA				
	Acc	F1	Pr	Rc	Time	Acc	F1	Pr	Rc	Time	Acc	F1	Pr	Rc	Time	Acc	F1	Pr	Rc	Time	Acc	F1	Pr	Rc	Time	Acc	F1	Pr	Rc	Time	Acc	F1	Pr	Rc	Time
Bot	0.56	0.55	0.82	0.56	0.003	0.97	0.97	0.97	0.97	0.030	0.96	0.96	0.96	0.96	0.014	0.97	0.97	0.97	0.97	0.008	0.98	0.98	0.98	0.98	0.170	0.69	0.62	0.61	0.69	0.125	0.68	0.68	0.84	0.68	0.003
DDoS	0.77	0.76	0.76	0.77	0.045	0.96	0.96	0.97	0.96	0.430	0.93	0.93	0.93	0.93	1.361	0.96	0.96	0.97	0.96	0.200	0.96	0.96	0.96	0.96	2.818	0.77	0.74	0.76	0.77	4.962	0.42	0.35	0.80	0.42	0.054
DoS GoldenEye	0.82	0.80	0.82	0.82	0.011	0.99	0.99	0.99	0.99	0.096	0.98	0.98	0.98	0.98	0.093	0.98	0.98	0.98	0.98	0.043	0.98	0.98	0.98	0.98	0.674	0.62	0.61	0.72	0.62	0.711	0.95	0.95	0.95	0.95	0.013
DoS Hulk	0.34	0.23	0.80	0.34	0.298	0.94	0.93	0.94	0.94	3.738	0.96	0.96	0.96	0.96	254.4	0.96	0.96	0.96	0.96	0.909	0.96	0.96	0.96	0.96	22.16	0.94	0.94	0.94	0.94	25.63	0.41	0.36	0.81	0.41	0.319
DoS Slowhttptest	0.41	0.36	0.73	0.41	0.006	0.98	0.98	0.98	0.98	0.056	0.99	0.99	0.99	0.99	0.058	0.98	0.98	0.99	0.98	0.020	0.99	0.99	0.99	0.99	0.313	0.72	0.71	0.83	0.72	0.387	0.42	0.38	0.74	0.42	0.006
DoS slowloris	0.42	0.36	0.80	0.42	0.006	0.95	0.94	0.94	0.94	0.055	0.95	0.95	0.95	0.95	0.035	0.96	0.96	0.96	0.96	0.022	0.95	0.95	0.95	0.95	0.372	0.77	0.76	0.80	0.77	0.494	0.48	0.46	0.79	0.48	0.008
FTP-Patator	1.00	1.00	1.00	1.00	0.006	1.00	1.00	1.00	1.00	0.057	1.00	1.00	1.00	1.00	0.214	1.00	1.00	1.00	1.00	0.014	1.00	1.00	1.00	1.00	0.412	1.00	1.00	1.00	1.00	2.683	1.00	1.00	1.00	1.00	0.008
Heartbleed	1.00	1.00	1.00	1.00	0.004	1.00	1.00	1.00	1.00	0.011	1.00	1.00	1.00	1.00	0.002	0.95	0.95	0.98	0.95	0.001	0.95	0.94	0.94	0.95	0.003	0.52	0.47	0.47	0.52	0.011	1.00	1.00	1.00	1.00	0.005
Infiltration	0.82	0.79	0.82	0.82	0.002	0.93	0.93	0.95	0.93	0.011	0.85	0.86	0.87	0.85	0.003	0.91	0.91	0.94	0.91	0.002	0.90	0.90	0.93	0.90	0.051	0.59	0.53	0.53	0.59	0.008	0.83	0.82	0.85	0.83	0.002
PortScan	0.44	0.39	0.80	0.44	0.185	1.00	1.00	1.00	1.00	2.554	1.00	1.00	1.00	1.00	54.72	1.00	1.00	1.00	1.00	0.784	1.00	1.00	1.00	1.00	15.01	0.72	0.61	0.63	0.72	13.77	0.84	0.84	0.89	0.84	0.205
SSH-Patator	0.41	0.34	0.80	0.41	0.008	0.96	0.96	0.96	0.96	0.069	0.96	0.96	0.96	0.96	0.045	0.96	0.96	0.97	0.96	0.027	0.96	0.96	0.97	0.96	0.411	0.87	0.87	0.88	0.87	0.324	0.47	0.43	0.80	0.47	0.006
Web Attack	0.73	0.75	0.86	0.74	0.005	0.97	0.97	0.97	0.97	0.029	0.93	0.94	0.94	0.94	0.014	0.96	0.96	0.96	0.96	0.009	0.97	0.96	0.96	0.97	0.170	0.69	0.64	0.68	0.69	0.111	0.83	0.84	0.89	0.84	0.003



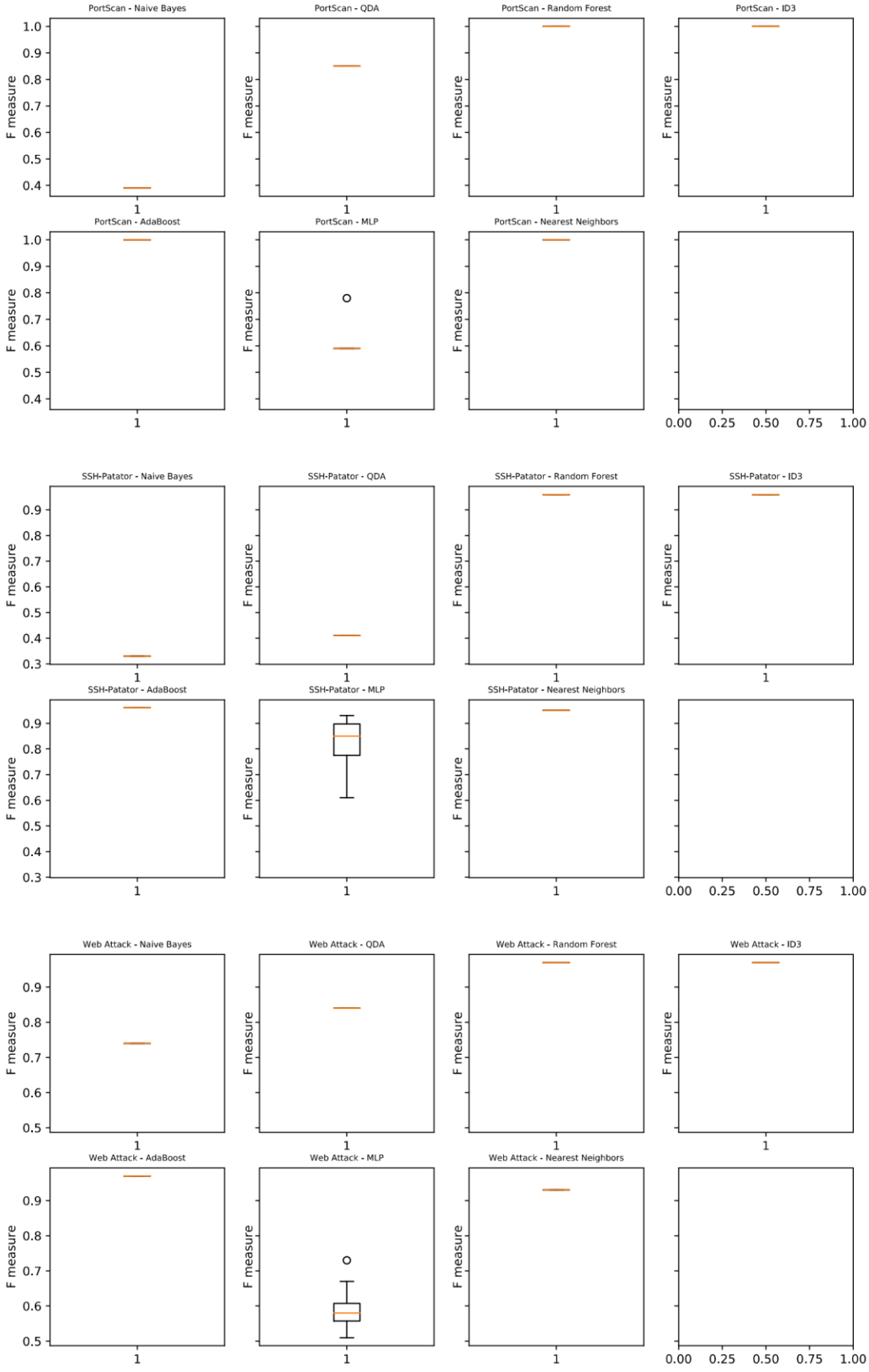
Appendix C Figure 2. Machine Learning Implementation Box and Whisker Graph (F-Measure)



Appendix C Figure 3. Machine Learning Implementation Box and Whisker Graph (F-Measure)

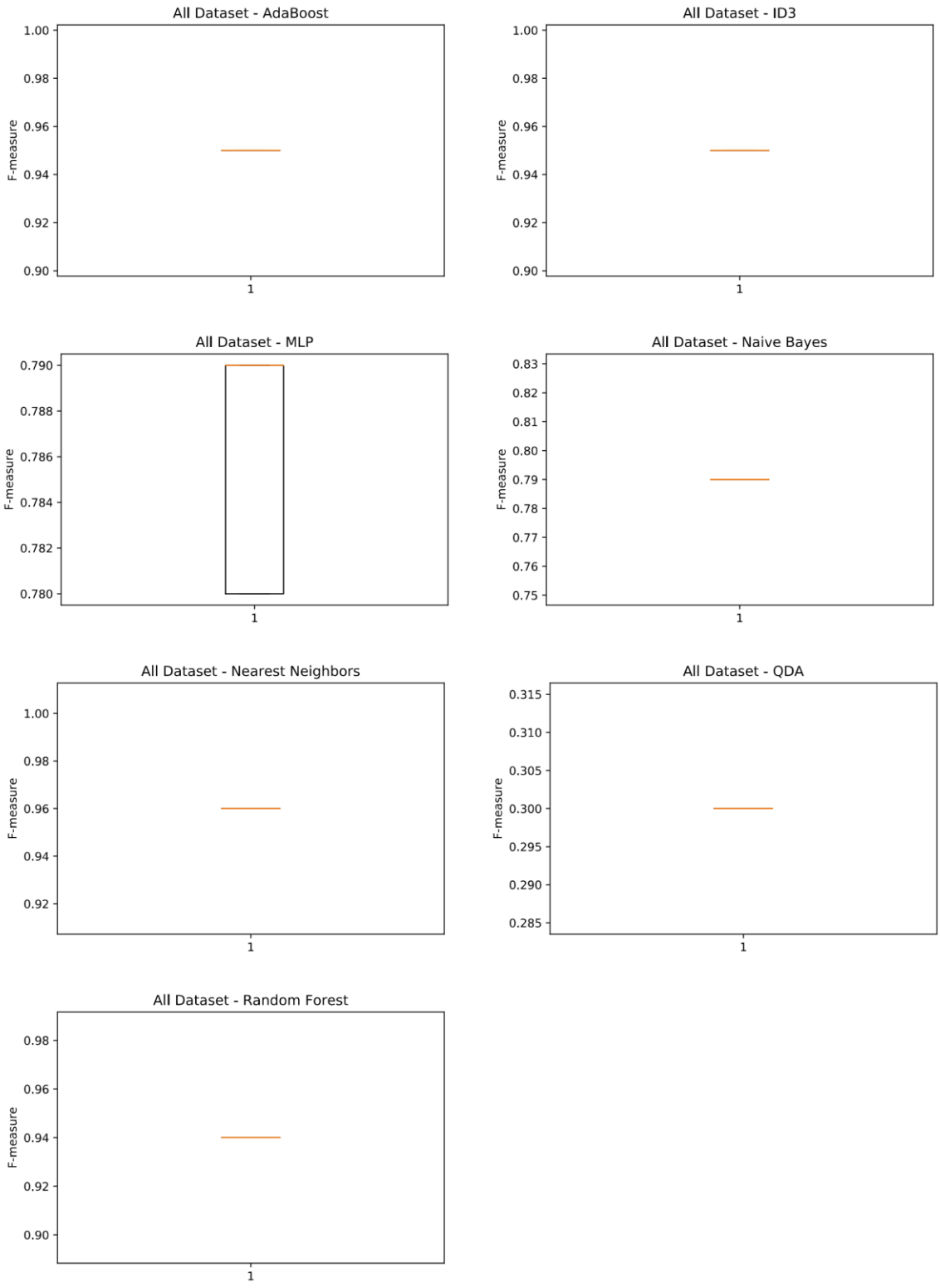


Appendix C Figure 4. Machine Learning Implementation Box and Whisker Graph (F-Measure)

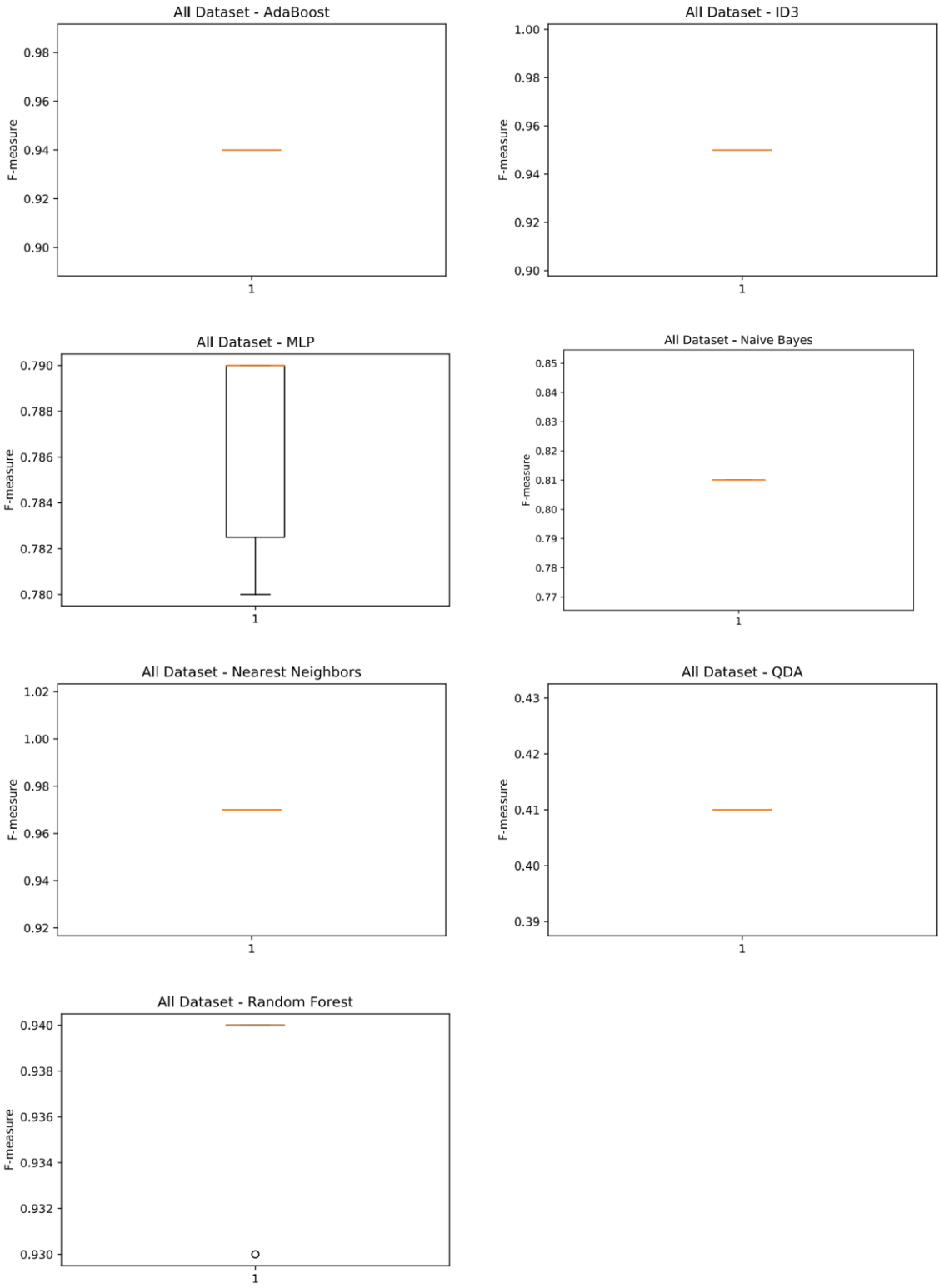


Appendix C Figure 5. Machine Learning Implementation Box and Whisker Graph (F-Measure)

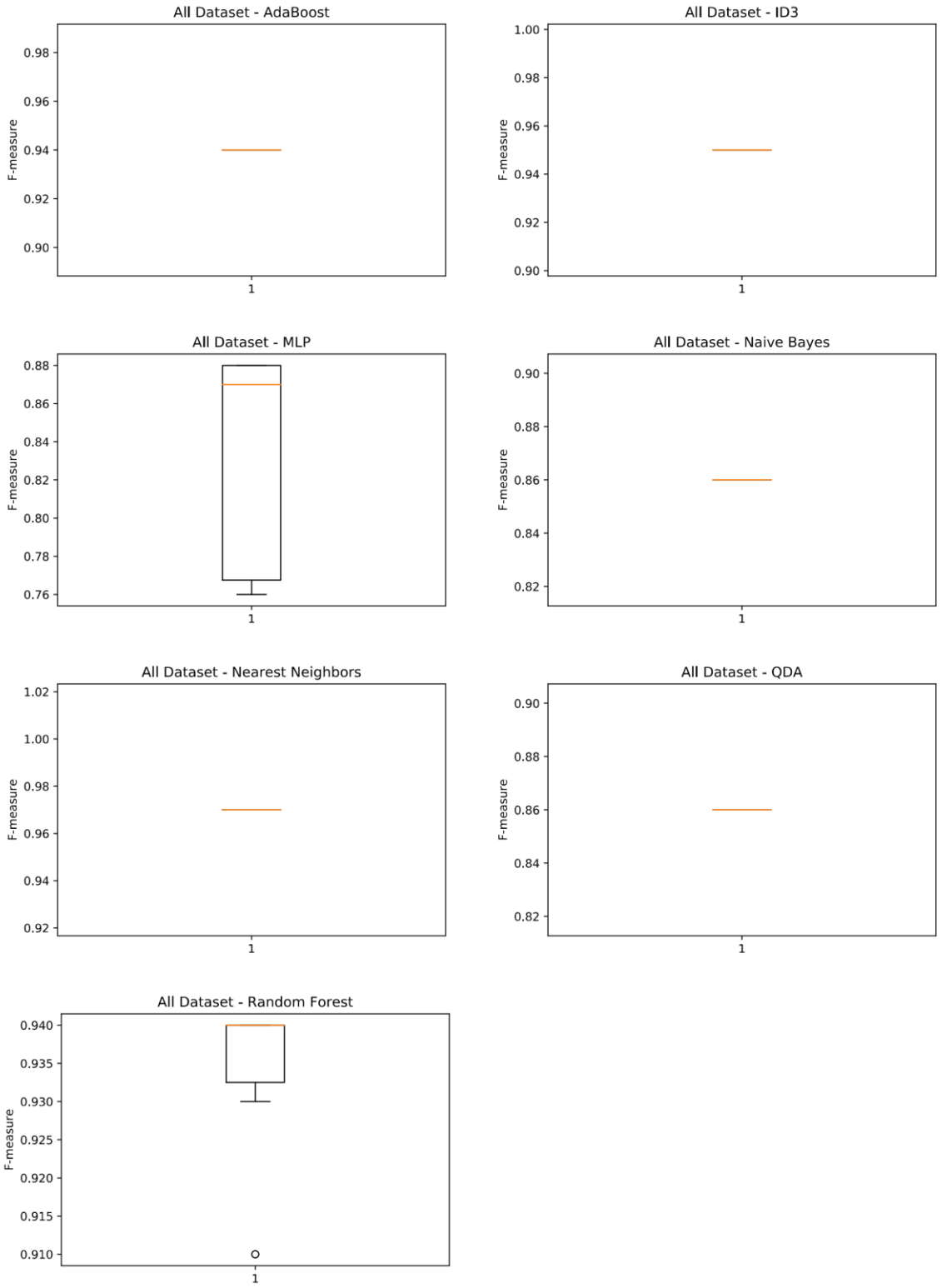
Appendix D. The Machine Learning Implementation Results (According to All Dataset)



Appendix D Figure 1. Method I - Machine Learning Implementation Box and Whisker Graph (F-Measure)



Appendix D Figure 2. Method II - Machine Learning Implementation Box and Whisker Graph (F-Measure)



Appendix D Figure 3. Final - Machine Learning Implementation Box and Whisker Graph (F-Measure)

Appendix E. Implementation "Readme" file

This file gives information on how to use the implementation files of "Anomaly Detection in Networks Using Machine Learning" (A thesis submitted for the degree of Master of Science in Computer Networks and Security written by Kahraman Kostas)

Python 3.6 was used to create the application files. Before running the files, it must be ensured that Python 3.6 and the following libraries are installed.

Library	Task
Sklearn	Machine Learning Library
Numpy	Mathematical Operations
Pandas	Data Analysis Tools
Matplotlib	Graphics and Visuality

The implementation phase consists of 5 steps, which are: 1- Pre-processing 2- Statistics 3- Attack Filtering 4- Feature Selection 5- Machine Learning Implementation

Each of these steps contains one or more Python files. The same file was saved with both "py" and "ipynb" extensions. The code they contain is exactly the same. The file with the ipynb extension has the advantage of saving the state of the last run of that file and the screen output.

Thus, screen output can be seen without re-running the files. Files with the ipynb extension can be run using the Jupyter notebook program. When running the codes, the sequence numbers in the filenames should be followed.

Because the output of almost every program is the prerequisite for the operation of the next program. Each step is described in detail below.

1 - Pre-processing

This step consists of a single file (preprocessing.ipynb). For this program to work, the dataset files must be in the "CSVs" folder in the same location as the program. The dataset files can be access here . (The reason that these files are given an external link is that the maximum limit of the file in the cseegit system is 10 MB).
As a result of executing this file, a file named "all_data.csv" is created. This file is a prerequisite for the other steps to work.

The most recent runtime of this file was recorded as 328 seconds. The technical specifications of the computer on which it is run are given below.

Central Processing Unit	:	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz
Random Access Memory	:	8 GB (7.74 GB usable)
Operating System	:	Windows 10 Pro 64-bit
Graphics Processing Unit	:	AMD Readon (TM) 530

2 - Statistics

This step consists of a single file (statistics.ipynb). This program examines the file "all_data.csv" and prints the statistics of attack and benign registry on this screen. It is not a prerequisite for any file. It only gives information. The last run time of this file was recorded as 13 seconds.

3 - Attack Filtering

This step consists of a single file (attack_filter.ipynb). This program uses the "all_data.csv" file to create attack files and then it saves them in the "./attacks/" location. The Dataset contains 12 attack types in total. Therefore, 12 CSV

files are created for these attacks. Within each file are 30% attack and 70% benign registry.This step is the prerequisite for the fourth and fifth steps. The last run time of this file was recorded as 304 seconds.

4 - Feature Selection

This step consists of two files.

a - feature_selection_for_attack_files.ipynb

This program uses attack files located under the "attacks" folder. The aim of this program is to determine which features are important for each attack. For this purpose, It is used the Random Forest Regressor algorithm to calculate the importance weights of the features in the dataset. These acquired features are used in machine learning section As a screen output, it sorts its features and weights from large to small and shows them on the bar chart (average 20 attributes per attack type).

The most recent run of this file was recorded as 4817 seconds.

b - feature_selection_for_all_data.ipynb

This program applies the previous step to the entire data set. Thus, it creates the feature importance weights of that is valid for the entire dataset. It uses the "all_data.csv" file and the Random Forest Regressor algorithm. As a screen output, it sorts its features and weights from large to small and shows them on the bar chart (20 attributes in total for all attacks).
The last run time of this file was recorded as 25929 seconds.

5 - Machine Learning Implementation

This step applies the machine learning algorithms to the data set and consists of 5 files.

a - machine_learning_implementation_for_attack_files.ipynb

this program uses the attack files under the "./attacks/" folder as a dataset. The features used are the 4 features with the highest weight for each file, produced by the feature_selection_for_attack_files file. This file applies 7 machine learning algorithms to each file 10 times and prints the results of these operations on the screen and in the file "./attacks/results_1.csv". It also creates box and whisker graphics of the results and prints them both on the screen and in the "./attacks/result_graph_1/" folder.
The last run time of this file was recorded as 3601 seconds.

b - machine_learning_implementation_with_18_feature.ipynb

This program implements machine learning methods in the file "all_data.csv". Uses the features used in the previous step. The set of features to be used consists of combining the 4 features with the highest importance-weight achieved for each attack in "machine_learning_implementation_for_attack_files" step under a single roof. Thus, 4 features are obtained from each of the 12 attack types, resulting in a pool of features consisting of 48 attributes. After the repetitions are removed, the number of features is 18.

This file applies 7 machine learning algorithms to "all_data.csv" file 10 times and prints the results of these operations on the screen and in the file "./attacks/results_2.csv". It also creates box and whisker graphics of the results and prints them both on the screen and in the "./attacks/result_graph_2/" folder.

The last run time of this file was recorded as 25082 seconds.

c - machine_learning_implementation_with_7_feature.ipynb

This program implements machine learning methods in the file "all_data.csv". The features used are the 7 features with the highest weight, produced by the feature_selection_for_all_data file. This file applies 7 machine learning algorithms to "all_data.csv" file 10 times and prints the results of these operations on the screen and in the file

"./attacks/results_3.csv". It also creates box and whisker graphics of the results and prints them both on the screen and in the "./attacks/result_graph_3/" folder.

The last run time of this file was recorded as 12714 seconds.

d - ml_f_measure_comparison.ipynb

This program runs with the file "all_data.csv". It finds feature giving the highest f-measure for Naive Bayes, QDA, and MLP algorithms, and prints them on the screen.

The last run time of this file was recorded as 2092 seconds.

e- machine_learning_implementation_final.ipynb

This program uses "all_data.csv" file as dataset. In feature selection, it follows a different path. To improve performance for the Naive Bayes, QDA and MLP algorithms, it uses the features generated by the ml_F-criterion_comparison file. In the other four algorithms, it uses 7 features with the highest significance, generated by the feature_selection_for_all_data file.

This file applies 7 machine learning algorithms to "all_data.csv" file 10 times and prints the results of these operations on the screen and in the file "./attacks/results_final.csv". It also creates box and whisker graphics of the results and prints them both on the screen and in the "./attacks/result_graph_final/" folder.

The last run time of this file was recorded as 18561 seconds.

Codes: https://github.com/bozbil/Anomaly-Detection-in-Networks-Using-Machine-Learning