Anomaly detection in networks

February 7, 2024

Abstract

Through the Internet, hundreds of thousands of institutions and millions of people exchange messages every day. As the ubiquity of networked systems continues to grow, the threat landscape for cybersecurity becomes increasingly sophisticated. Anomaly detection plays a pivotal role in fortifying networks against malicious activities by identifying deviations from normal behaviour. This paper presents a comprehensive review of anomaly detection techniques in network environments, emphasizing their significance in mitigating cyber threats.

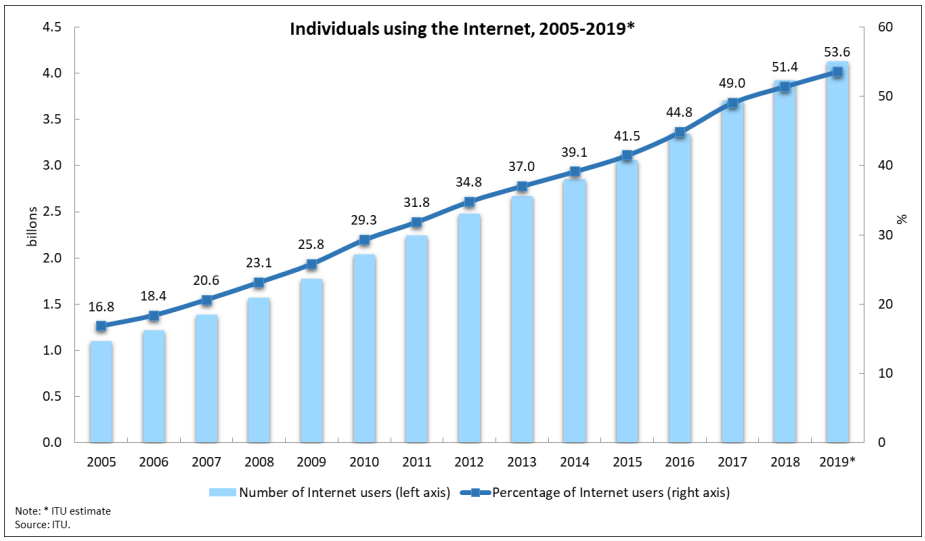
The review encompasses traditional and emerging anomaly detection methods, including statistical approaches, machine learning algorithms, and deep learning models. We delve into the challenges associated with anomaly detection, such as the dynamic nature of network behaviour, evolving attack strategies, and the need for real-time analysis. Furthermore, we explore the impact of data imbalances, scalability concerns, and interpretability issues on the effectiveness of anomaly detection systems.

Contents

1 Introduction

1.1 Motivation

Through the Internet, hundreds of thousands of institutions and millions of people communicate with one another every day. The number of individuals utilizing the Internet has grown significantly over the past 20 years; currently, there are over 4 billion users, and the growth is still going strong.



In tandem with these advancements, there is a daily rise in the quantity of cyber-attacks. To protect information security against these assaults, two fundamental techniques are employed for attack detection: signature-based identification and anomaly-based detection.

The database that signature-based techniques developed is used to identify assaults. This approach works rather well, but it requires ongoing database updates and processing of fresh attack data. Furthermore, even current databases are susceptible to zero-day attacks, which are assaults that have never been observed before. They are unable to stop these assaults since they are not included in the database. The goal of the anomaly-based method is to identify anomalous network behaviours by analyzing network flow. This technique works well against zero-day attacks since it has demonstrated success in identifying assaults that it has never experienced before.

Furthermore, the SSL / TLS (Secure Sockets Layer / Transport Layer Security) protocols are used to encrypt more than half of today's internet traffic, and this percentage is rising daily. When dealing with encrypted internet streams, signature-based techniques are ineffective due to the inability to view the contents of the stream.

On the other hand, data is analyzed using generic attributes like size, packet count, and connection duration in the anomaly-based method. For this reason, it can analyze encrypted protocols without needing to view the content of the messages themselves. The anomaly-based detection technique is widely used to identify and stop network assaults because of all these benefits.

The goal of this study was to add to the body of literature by creating a machine learning-based system that can quickly and accurately identify network anomalies.

1.2 Goals and objectives

1.2.1 Goals

After this study, the following objectives are sought to be fulfilled:

1. Analyzing machine learning techniques to identify abnormalities in networks.
2. To use machine learning techniques to examine network anomalies and quickly and effectively detect network threats.
3. To assess the study's degree of success by contrasting its findings with those of earlier research projects in this field.
4. Adding to the body of literature by closely examining findings from earlier research on the identification of network abnormalities.

1.2.2 Objectives

1. To perform in-depth field research to review earlier work on the topic.
2. Deciding which dataset is best by thoroughly investigating the alternatives to the dataset.
3. Selecting appropriate algorithms through in-depth machine learning algorithm study.
4. Selecting the appropriate algorithms through a thorough investigation of machine learning techniques.
5. Deciding on the right software platform.
6. Selecting the appropriate platform for hardware and equipment.
7. Selecting appropriate assessment standards.
8. Selecting the reference studies with which to compare throughout the assessment stage

2 Datasets and background

2.1 Datasets

Large volumes of both malicious and benign network traffic are required for training and testing phases in the machine learning process of detecting network anomalies. Real network traffic, however, cannot be exploited for public purposes due to privacy concerns. Numerous datasets have been and will continue to be developed to address this demand. This step will provide an overview of a few well-known datasets. These will then be compared and assessed to determine which ones to employ in the implementation stage.

2.1.1 ISCX 2012

Even though a large number of datasets have been utilized to identify anomalies, concerns have been raised because the synthetic data used or the outdatedness of the original datasets have not been accurately reflected. To address these issues, the Canadian Institute for Cybersecurity's testbed, which consists of a seven-day Internet stream, was used to develop the ISCX 2012 (Intrusion detection evaluation dataset) in 2012. Some dataset's standout features are:

1. FTP, HTTP, IMAP, POP3, SMTP, and SSH protocols, as well as actual regular and malicious streams, were produced since it was designed utilizing real devices.
2. All data are labelled
3. There is a lot of variation in the attacks, which include Brute Force SSH, Distributed Denial of Service, Denial of Service, and Infiltration.

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

However, it appears that the ISCX 2012 data set will not be sufficient to fulfil current demands since it excludes SSL/TLS (Traffic Sockets Layer/Transport Layer Security) traffic, which makes up more than half of Internet traffic today.

2.1.2 DARPA 98

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).

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). 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

2.1.3 KDD 99

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. 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

2.1.4 CAIDA

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. 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

2.1.5 NSL-KDD

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 NSLKDD dataset consists of 4 parts under two main headings as training and testing: training data (KDDTrain+), 20

$$smallerversionofthetestdatawithalldifficultylevels(KDDTest - 21$$

2.1.6 CICIDS 2017

CICIDS 2017, or the Canadian Institute for Cybersecurity Intrusion Detection Systems (CICIDS) 2017 dataset, is a publicly available dataset that is commonly used in the field of cybersecurity for evaluating intrusion detection systems (IDS). The dataset was created by the Canadian Institute for Cybersecurity and includes network traffic data for various types of cyber attacks.
Generating realistic background traffic was the author's top priority in building this dataset. They have used their proposed B-Profile system (Sharafaldin, et al. 2016) to profile the abstract behaviour of human interactions and generate naturalistic benign background traffic. For this dataset, they built the abstract behaviour of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols.
The data capturing period started at 9 a.m., Monday, July 3, 2017, and ended at 5 p.m. on Friday, July 7, 2017, for a total of 5 days. Monday is the normal day and only includes the benign traffic. The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. They have been executed both morning and afternoon on Tuesday, Wednesday, Thursday and Friday.

№	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

№	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

2.1.7 Advantages of CICIDS 2017

1. The obtained data is real-world data; was obtained from a testbed consisting of real computers

2. Data streams are collected from computers with an up-to-date operating system. There is operating system diversity (Mac, Windows, and Linux) between both attacker and victim computers.

3. Data sets are labelled. In order to apply the machine learning methods, feature extraction, which is a critical step, was applied and 85 features (see Appendix A for the feature list) were obtained.

4. Both raw data (pcap files - captured network packets files) and processed data (CSV files-comma-separated data files) are available to work on.

5. 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.

6. 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.

2.1.8 Limitations of CICIDS 2017

1. Raw data files and processed data files are very large (47.9 GB and 1147.3 MB respectively).

2. 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.

3. 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.

2.2 Anomaly in network traffic

2.2.1 Anomaly Types

A sample that lacks clearly defined characteristics of a normal sample is called an anomaly. The norms that comprise the normal idea need to be precise and legitimate in order for the abnormality to be comprehended. Three categories are used to analyze the phenomenon:

1. Point anomaly: A point anomaly is a data sample that differs from the rest of the dataset in terms of its attributes. For instance, a person who spends little on credit cards on a regular basis may spend a lot on a single day. This is considered an abnormality.

2. Contextual anomaly: Out-of-pattern behaviour in a data sample might be triggered by specific situations. For instance, a person who uses credit cards sparingly on a daily basis may spend significantly more on a feast day.

3. Collective anomaly: A collective anomaly occurs when a collection of comparable data has anomalous features compared to normal data (11). For example, the surge in credit card spending over Valentine's Day is a collective oddity.

2.2.2 Network Attacks Types

Network security protects against assaults on three principles: confidentiality, integrity, and availability:

1. Confidentiality: Access to information should be limited to authorized users and prevent unauthorized access.

2. Integrity: Only the genuine user can add, amend, or delete information. Unauthorized individuals should be unable to edit information.

3. Accessibility: The system should always be available to legitimate users.

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

1. Denial of Service (DoS): This sort of attack involves an attacker abusing system resources to prevent genuine users from using the service. Sending a huge amount of queries to a web server might cause it to drop out of service. Dos assaults may be classified into two types: bandwidth and resource depletion. Bandwidth depletion attacks use high data flow to deplete a victim's bandwidth, whereas resource depletion attacks use many packages to eat memory and processing resources.

2. Probe (Information Gathering): These assaults aim at obtaining information about the victim. This technique provides attackers with valuable information, including network structure, operating system, device kinds, and attributes. Although this assault does not immediately impact the system, it is crucial since it lays the groundwork for future attacks that might hurt it.

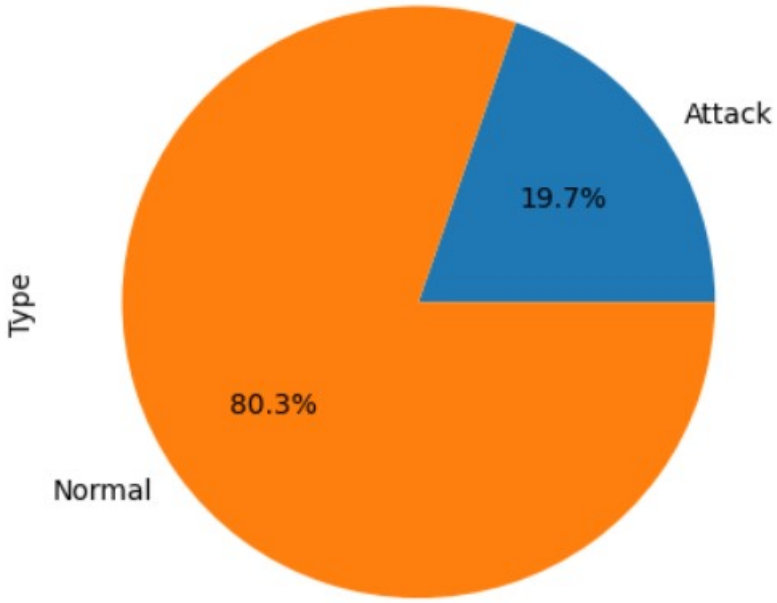
3. U2R (User to Root): This sort of attack involves an attacker attempting to get control of the administrative account to steal valuable resources. An attacker may acquire access to the Administrator account by exploiting system vulnerabilities or via brute-force assaults.

4. R2U / R2L (Remote to User / Remote to Local): This attack involves infiltrating the victim's network and gaining the ability to transmit packets from their machine. An attacker can achieve this privilege by exploiting system vulnerabilities or via brute force assaults.

Classifying network attacks based on their resulting abnormalities can aid in detection. Each assault generates a variation (an anomaly) in the network flow. DoS attacks can increase network traffic and packet volume. During a DoS attack, several irregular streams may be stated. Thus, DoS assaults should be classified as collective abnormalities. U2R and R2U attacks should be classified as contextual and point anomalies, as they target certain users, ports, and purposes. The probing attack increases network traffic and serves a specific objective. In this context, the probe attack can be defined as a contextual and collective anomaly

2.3 Types of attack in dataset

In this section, the types of attacks that the dataset contains are examined in detail. The dataset has 15 labels for all of its data. One tag (Benign) indicates regular network movements, while the remaining 14 indicate assaults. The benign record, generated using Mail, SSH, FTP, HTTP, and HTTPS protocols, simulates genuine user data and is not hazardous to the network. The figure below shows the names and numbers of these labels.



Examining the amount of attacks, it's clear that some occur often. The DoS HULK assault accounts for over half of all attacks, while the PortScan attack makes for one-third. The nature of the attacks contributes to the imbalance in distribution. DoS and PortScan attacks produce excessive data and packet flows. During these attacks, it's common to see more traffic compared to usual usage and other forms of attacks.

- The attacks in the dataset can be explained as follows:

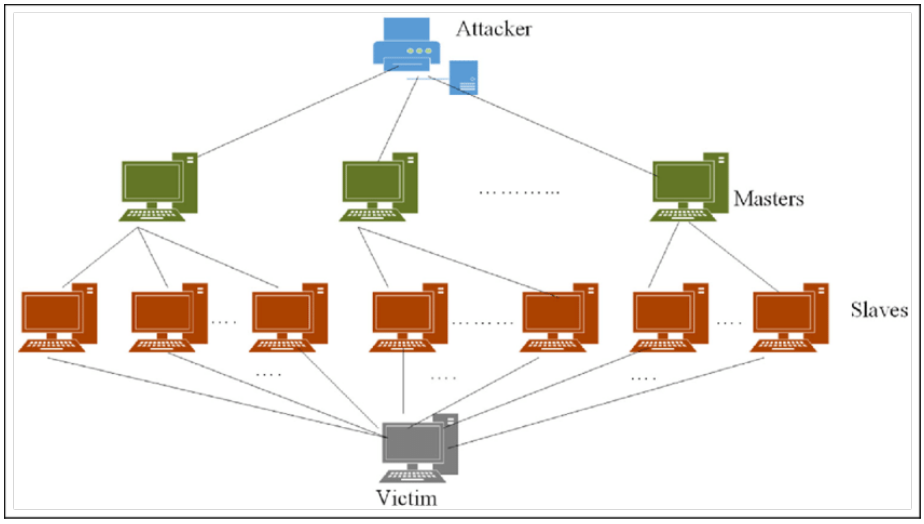
1. DoS HULK: The CICIDS2017 data set shows that HULK (HTTP (Hypertext Transfer Protocol) Unbearable Load King) occurs on Wednesdays, as do other DoS assaults. HULK may be used for DoS and DDoS assaults. This attack can quickly stop a server by directly affecting its load. HULK

may mask the user agent and utilize unique templates for each attack request. During the attack, it generates TCP-SYN floods and numerous HTTP-GET flood requests. Readers can examine TCP-SYN and HTTP-GET flood techniques to gain insight into this assault.

2. PortScan: The PortScan attack in the CICIDS2017 dataset is carried out by Nmap and may be found in the Friday afternoon portion. Ports are connection points with numbers ranging from 0 to 65535 that specify the sort of service to receive from the computer. PortScan is a prominent attack type that gathers information on how attackers might get access to the system. This attack provides vital information about linked devices, including operating systems, services, and port status. A PortScan attack involves the attacker sending a message to each port. The victim's answer indicates if the port is used or not. Port Scanning types can be listed as follows: SYN Scan, TCP Connect Scan, ACK Scan, FIN Scan, NULL Scan, XMAS Scan, UDP Scan, Fragmentation Attack

3. DDoS: The DDoS assault in the CICIDS2017 dataset is carried out by the application LOIC, which sends HTTP, TCP, and UDP queries. is placed in the Friday-afternoon section. A DDoS (Distributed Denial-of-Service) assault, like a DoS attack, aims to prohibit a genuine user from accessing a website or online service. Unlike DoS attacks, DDoS attacks originate from several computers and target a single source.

   Hundreds of proxy machines can be deployed to attack one computer. These machines were infected with malicious software, allowing them to be utilized remotely during the DDoS assault. These devices are referred to as "zombies," and the network they form is known as a "botnet".



4. DoS Goldeneye: DoS Goldeneye takes place on Wednesday-records with other DoS attacks within the CICIDS2017 dataset. It is a Python-based denial-of-service attack. The purpose of this attack is to drain the victim's system resources, preventing genuine users from obtaining service. Goldeneye is a multithreaded attack that effectively uses CPU and memory hardware to perform an HTTP Flood attack.

   The attack does not encrypt packets or generate a phony source IP address (IP spoofing). It is compatible with all Linux, Mac, and Windows operating systems.

   The Keep-Alive approach optimizes the file size sent over a single TCP connection. Also, the No-Cashed message feature disables HTTP-Cache-Control. Using these two functionalities consumes system resources fast.

5. FTP-Patator: FTP-Patator attack is made using the Patator, a multithreaded tool written in the Python program. This attack takes place on Tuesday-morning-records within the CICIDS2017 dataset.

   FTP (File Transfer Protocol) allows for file transfers between clients and servers over a network. To transmit a file over FTP, you must have a valid account and password on the network. The FTP-Patator attack uses brute force to get the username and password.

   Brute-force is a cryptography attack that aims to get passwords. The attacker attempts to get in as a legitimate user by using several usernames and passwords. This method is frequently carried out using password-generating software. Passwords are less effective against brute-force assaults due to user preference for memorable and meaningful passwords.

   Brute force assaults often include several failed access attempts in a short period of time. Dense packet flow can be observed during a brute-force assault. Additionally, unsuccessful entries do not include large files, resulting in minimal bandwidth use and bytes count.

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

   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.

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: Scanning Step, Brute-Force Step, Die-off Step

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

7. DoS Slowloris: DoS Slowloris takes place on Wednesday-records with other DoS attacks within the CICIDS2017 dataset. 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 and aims to reduce the number of users that the server can serve. 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

8. DoS SlowHTTPTest:DoS SlowHTTPTest takes place on Wednesday-records with other DoS attacks within the CICIDS2017 dataset. 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.

   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.

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. However, connections that have window sizes that are much smaller than normal flow will give you a hint about this attack.

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

   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.

   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.

10. Web attack: Web Attack takes place on Thursday-afternoon -records in the CICIDS2017 dataset. In this group of attacks, three different web-based attacks were launched against a vulnerable PHP (Hypertext Preprocessor - a script programming language)/ MySQL (an open source database administration framework) web application identified as the victim. These attacks are: Brute Force, XSS (Cross-Site Scripting), SQL (Structured Query Language – a database management system) Injection.

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.

    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.

12. Heartbleed: This attack is made using the Heartleech, a Heartbleed exploit tool written in C programming language. This attack takes place on

Wednesday - afternoon -records within the CICIDS2017 dataset. 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.

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.

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.

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 16 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.

3 Methodology

3.1 Machine learning

Machine learning is a science and art that enables the programmed computers to learn from the data given to them. 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:

1. It can interpret very large and complicated data.
2. It can solve complex problems that traditional methods cannot find solutions.
3. Machine Learning methods can find solutions to situations where existing solutions require too much external intervention/update without external

intervention.

4. 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.

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. 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. 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. This method combines the high performance of supervised learning with the low cost of unsupervised learning. 17 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.

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.

3.1.1 Naïve Bayes

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

Bayes' theorem is expressed by the following equation:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

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.

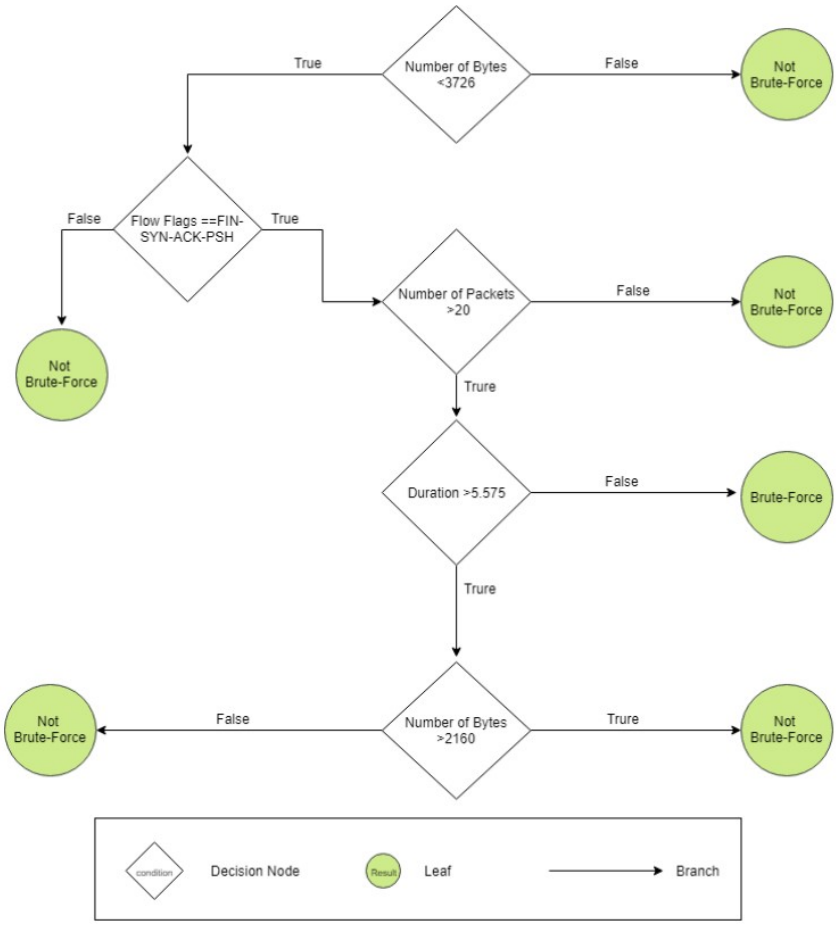
In this statistical network, it is assumed that the child nodes are independent of each other. Although this hypothesis is the basis of Naive Bayes' theory, the assumption that sub nodes are independent of one another is often not correct. This is the underlying reason for the Naive Bayes method to achieve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

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.

3.1.2 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 18 two in some sub-algorithms). This selected branch takes the algorithm to the next node. This process ends with the last element, the leaf.



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.

Unfortunately, these complex trees lead to overfitting2 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. 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.

3.1.3 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.

Random forest generates N decision trees from a 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.

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

1. This algorithm can work well with very large and complicated datasets.
2. The overfitting problem frequently encountered by decision trees is very rare in this algorithm.
3. It can be applied to a lot of kinds of machine learning problems.
4. It is also good to deal with missing values in the data set. It replaces these lost values with their own created values.
5. 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 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.

3.1.4 K-nearest neighbor

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.

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

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).

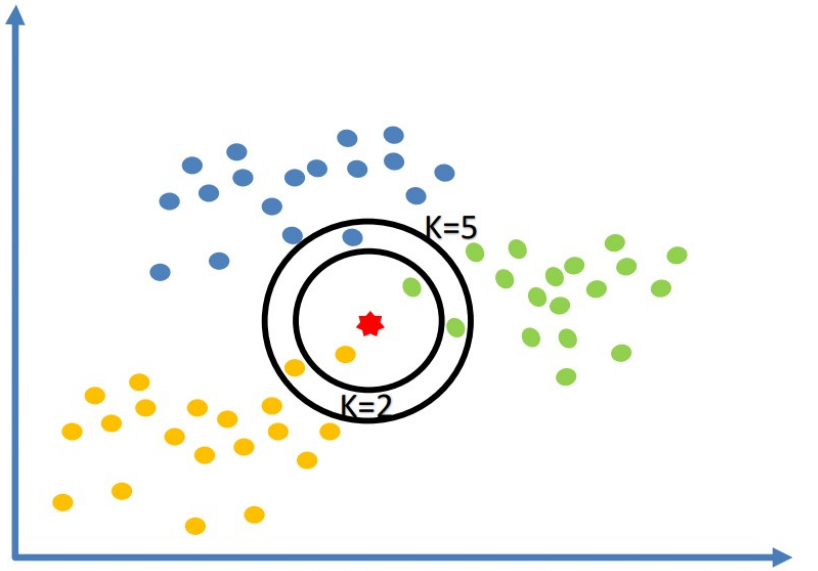


Figure 8. Operation of KNN algorithm for K = 2 and K = 5 values.

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

3.1.5 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.

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.

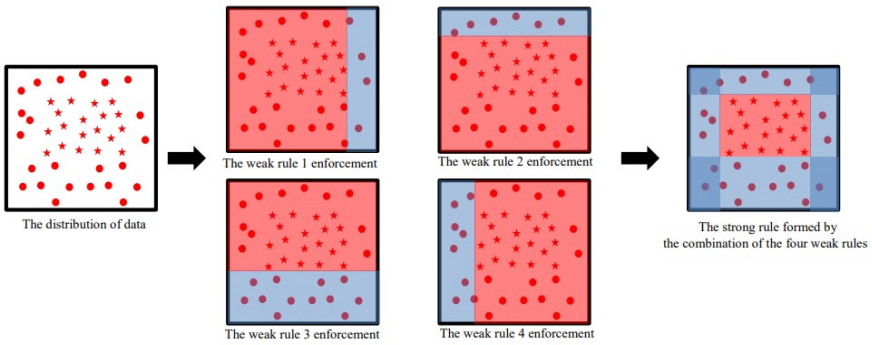


Figure 9. Demonstration of the operation of the AdaBoost algorithm.

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

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

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.

3.1.6 Multilayer Perceptron

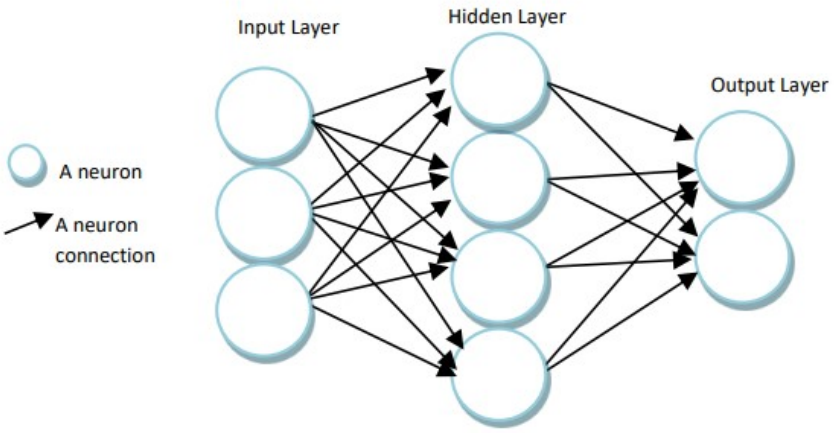
MLP (Multi-Layer Perceptron) is a genre of artificial neural networks. Artificial neural networks (ANN) is a machine learning method that takes inspiration from the way the human brain works. The intention of this method is to imitate the properties of the human brain, such as learning, decision making, 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.

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.

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.

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.



The advantages MLP provides are as follows:

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

But from the other side:

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

3.1.7 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.

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.

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

3.2 Platform background

3.2.1 Main Software Platform

Python, 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, Python3.9 has been chosen to be used in this work, because of many of the advantages it provides.

Sklearn (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 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 is a library that runs on Python, allowing visualization of data. This library is used to create graphs used in the study.

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

3.2.2 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:

This is a platform on Google Colab (Python 3 Google compute engine backend), some main properties could be understood as follow:

1. 12.7 GB ram
2. T4 GPU

4 Models evaluation

4.1 Performance Evaluation Metrics

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.

$$Accuracy = \frac{TN + TP}{TP + FP + TN + FN}$$

Recall (Sensitivity): Recall measures the proportion of actual positives that were predicted correctly. It is the True Positive out of all Ground Truths. The ratio of data classified as an attack to all attack data. Mathematically, it is defined as follows.

$$R = \frac{TP}{TP + FN} = \frac{TP}{TotalGroundTruths}$$

Precision: The ratio of successful classified data as the attack to all data classified as the attack

$$Precision = \frac{TP}{TP + FP}$$

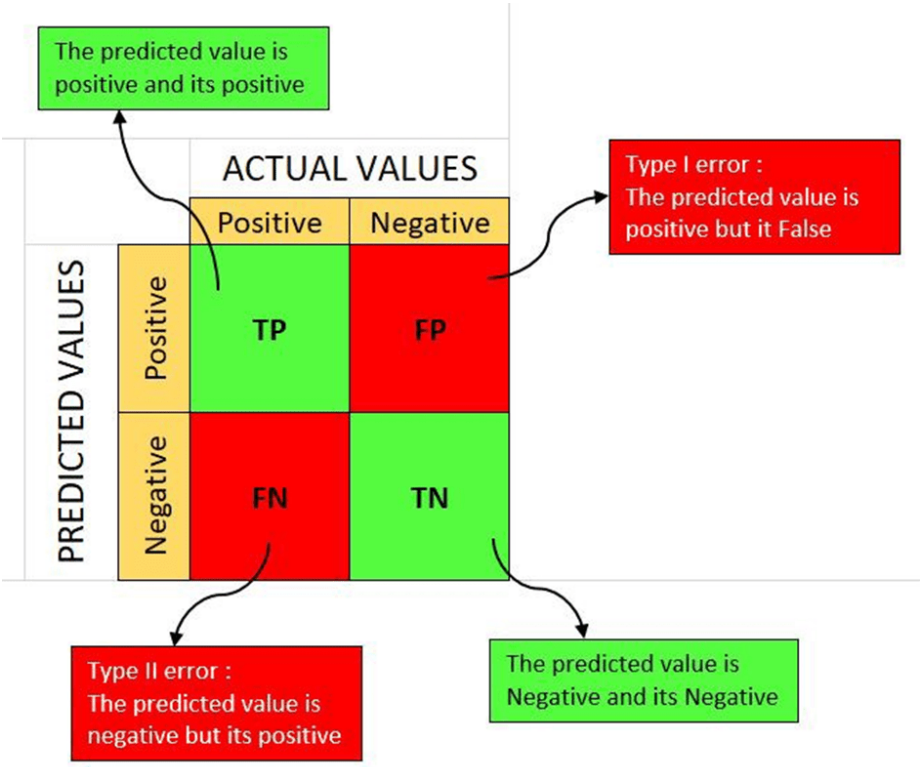
F-measure (F-score/F1-score): The harmonic-mean of sensitivity and precision. This concept is used to express the overall success. so, in this study, when analysing the results, it will be focused, especially on the F1 Score

$$F - Measure = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

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

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

This distribution is presented by visualizing Confusion matrix in Figure below.



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.

4.2 Result

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. 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.

Machine Learning Algorithms	Evaluation Criteria				
	Accuracy	Precision	Recall	F-score	Time
Naive Bayes	0.58	0.7	0.58	0.51	1.742
QDA	0.86	0.88	0.86	0.86	4.6375
Random forest	0.95	0.95	0.95	0.95	5.7898
ID3	0.97	0.97	0.97	0.97	11.3384
AdaBoost	0.99	0.99	0.99	0.99	149.3572
MLP	0.74	0.83	0.74	0.73	237.2962
K Nearest Neighbours	0.99	0.99	0.99	0.99	2167.6047

5 Conclusion and Future Work

5.1 Conclusion

The goal of this project is to use machine learning to detect network anomalies. The CICIDS2017 dataset was chosen because to its up-to-dateness, diverse attacks, and support for numerous network protocols (e.g., mail services, SSH, FTP, HTTP, and HTTPS). This dataset has nearly 80 characteristics that characterize the network flow. The Random Forest Regressor technique was used to calculate significance weights and choose features for machine learning. These computations were performed using two ways. In the first place, importance weights are determined for each assault type. The second technique collects all assaults into a single group and calculates their significance weights. That is, the common qualities that are relevant to all assaults are identified. The data was analyzed using seven different machine learning methods.

5.2 Future work

This work is open to further enhancements. This section will highlight some potential areas for improvement. This study employed CSV files providing network flow characteristics for training and testing. Unfortunately, this strategy is not practicable for real-world systems. To address this issue, build a module that captures actual network data and integrates it with machine learning algorithms. This study used many machine learning approaches separately, yielding experimental findings. However, this approach has limited practical applications in real life. To address this issue, a multilayered/hierarchical machine learning system can be created. Furthermore, this structure allows for time, CPU power, and memory savings. In a two-tiered topology, using quick and computationally inexpensive algorithms like Naive Bayes or QDA on the first layer allows for continuous and cost-effective network traffic monitoring. The initial stage detects anomalies and sends them to a higher-performance layer of algorithms. This layer measures by creating a decision mechanism. When an

anomaly is detected, the initial step sends it to a higher-performing layer that includes algorithms like ID3, AdaBoost, and KNN. The last layer of the determination procedure takes precautionary measures to safeguard the network against attacks.

6 References

1. R. E. Schapire, "The boosting approach to machine learning: An overview," in Nonlinear estimation and classification: Springer, 2003, pp. 149-171.

2. E. Alpaydin, Introduction to machine learning. MIT press, 2009.

3. 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.

4. 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.

5. 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.

6. 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.

7. S. Mukherjee and N. Sharma, "Intrusion detection using naive Bayes classifier with feature reduction," Procedia Technology, vol. 4, pp. 119-128, 2012.

8. 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.

9. "Python 3.9 documentation," Python Software Foundation. Available: https://docs.python.org/3.6/.

10. "scikit-learn," scikit-learn 0.19.1 documentation. [Online]. Available: http://scikit-learn.org/stable/.

11. "Python Data Analysis Library," pandas: powerful Python data analysis toolkit - pandas 0.22.0 documentation. [Online]. Available: https://pandas.pydata.org/.

12. "Matplotlib: Python plotting - Matplotlib 2.2.2 documentation," Matplotlib. [Online]. Available: https://matplotlib.org/.

13. “NumPy,” NumPy developers. [Online]. Available: http://www.numpy.org/.

14. 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.

15. “sklearn.preprocessing.LabelEncoder,” 1.4. Support Vector Machines - scikit-learn 0.19.1 documentation. [Online]. Available: http://sciki-learn.org/stable/modules/generated/s

16. 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/.

17. 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.