PREDICTIVE MODELING FOR BOTNET DETECTION: A NEW DATASET AND MACHINE LEARNING APPROACH

Kadir İlker BUDAK

AUGUST 2023

ÇANKAYA UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

M.Sc. Thesis in

COMPUTER ENGINEERING

PREDICTIVE MODELING FOR BOTNET DETECTION: A NEW DATASET AND MACHINE LEARNING APPROACH

Kadir İlker BUDAK

AUGUST 2023

ABSTRACT

PREDICTIVE MODELING FOR BOTNET DETECTION: A NEW DATASET AND MACHINE LEARNING APPROACH

BUDAK, Kadir İlker

M.Sc. in Computer Engineering

Supervisor: Assist. Prof. Dr. Ayşe Nurdan SARAN

August 2023, 74 pages

With the development of technology, the importance of online services has gradually increased. By managing the zombie network consisting of botnets, the attackers target the slowdown or interruption of the services by making more requests to the systems and networks than their capacity. This type of attack is called DDOS (Distributed Denial of Service attack). A literature study has been conducted to detect and prevent DDOS attacks and many different techniques have been encountered. As a result of the research, DDOS behavior detection has been focused on by using machine learning. In this study for behavior-based DDOS detection with Machine Learning, the CTU-13 and the virtual dataset created in the local environment were used. The data sets have been made ready for study by applying normalization processes. 5 different algorithms have been used for Machine Learning and parameter tuning has been performed on the algorithms. The effect of different methods, such as multiple regression, stacking method, and feature diversity on the result has been evaluated.

The effects of the improvements on the results are discussed. In general, Random Forest and Decision Tree stand out as successful algorithms. Naive Bayes and Support Vector Machine have been unsuccessful for the case studied. In the stacking method, the two algorithms working together have positively affected the result.

The most important result of the virtual dataset is that using the IP address feature does not positively contribute to the result.

Keywords: DDOS, DDOS Behavior, CTU-13 Dataset, Machine Learning Algorithms, DDOS Prevention, Botnet Detection

ÖZ

BOTNET TESPİTİ İÇİN TAHMİN MODELİ: YENİ BİR VERİ SETİ VE MAKİNE ÖĞRENME YAKLAŞIMI

BUDAK, Kadir İlker
Bilgisayar Mühendisliği Yüksek Lisans

Danışman: Dr. Öğr. Üyesi Ayşe Nurdan SARAN
Ağustos 2023, 74 sayfa

Teknolojinin gelişmesiyle birlikte çevrimiçi hizmetlerin önemi giderek artmıştır. Saldırganlar, botnet'lerden oluşan zombi ağını yöneterek, sistem ve ağlara kapasitelerinden fazla istekte bulunur, böylece hizmetlerin yavaşlamasını veya kesintiye uğramasını hedefler. Bu tür saldırılara DDOS (Dağıtılmış Hizmet Reddi saldırısı) adı verilir. DDOS saldırılarının tespiti ve önlenmesi için literatür çalışması yapılmış ve birçok farklı teknikle karşılaşılmıştır. Yapılan araştırmalar sonucunda Makine Öğrenmesi ve DDOS davranış tespiti konularına ağırlık verilmiştir. Machine Learning ile Davranış tabanlı DDOS tespiti için yapılan bu çalışmada, CTU-13 veri seti ve yerel ortamda oluşturulan sanal veri seti kullanılmıştır. Veri setleri üzerine normalizasyon işlemleri uygulanarak çalışma için hazır hale getirilmiştir. Makine Öğrenemsi için 5 farklı algoritma kullanılmış ve algoritmalar üzerinde parametre ayarı yapılmıştır. Çoklu regresyon, topluluk öğrenimi ve özellik çeşitleme gibi farklı yöntemlerin sonuca etkisi değerlendirilmiştir.

İyileştirmelerin sonuçlar üzerindeki etkileri tartışılmıştır. Genel olarak Random Forest ve Decision Tree başarılı algoritmalar olarak öne çıkmaktadır. Naive Bayes ve Support Vector Machine, bu senaryo için başarısız olmuştur. Topluluk öğrenim yönteminde birlikte çalışan iki algoritmanın sonuca olumlu etkisi olmuştur. Sanal veri setinin en önemli sonucu, ip adresi özelliğinin kullanımının sonuca olumlu bir katkısının olmamasıdır.

Anahtar Kelimeler: DDOS, DDOS Davranıı, CTU-13 Veriseti, Makine Öğrenmesi Algoritmaları, DDOS Koruması, Botnet Tespiti

ACKNOWLEDGEMENT

First of all, I would like to express my sincere gratitude to my family, especially my wife, for their encouragement, support and the sacrifices to me.

Special thanks to my supervisor Assist. Prof. Dr. Ayşe Nurdan SARAN for the excellent guidance and providing me with an excellent atmosphere to conduct this research. My special gratitude also goes to the rest of the thesis committee Assist. Prof. Dr. Ulaş GÜLEÇ and Assist. Prof. Dr. Abdül Kadir GÖRÜR for the encouragement and insightful comments

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURE

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOLS

w	: Weight Vector (θ1)
x	: Input Vector
b	: Deviation (θ0)
g	: Gamma Constant
c	: Constant c0
d	: Degree of the Kernel

ABBREVIATIONS

DOS	: Denial of Service
DDOS	: Distributed Denial of Service
OSI	: Open Systems Interconnection
TCP	: Transmission Control Protocol
UDP	: User Datagram Protocol
IP	: Internet Protocol
ICMP	: Internet Control Message Protocol
DNS	: Domain Name System
NTP	: Network Time Protocol
SNMP	: Simple Network Management Protocol
SSDP	: Simple Service Discovery Protocol
C&C	: Command and Control
ML	: Machine Learning
K-NN	: K-Nearest Neighbor
SVM	: Support Vector Machine
NB	: Naive Bayes
DT	: Decision Tree
USML	: Unsupervised Learning
LR	: Logistic Regression

MLP	: Multilayer Perceptron
SDN	: Software-Defined Networking
DNN	: Deep Neural Network
RF	: Random Forest
MAC	: Media Access Control Address
TRP	: True Positive Rate
FRP	: False Positive Rate
ANN	: Artificial Neural Network
AUC	: Area Under the ROC Curve
MCC	: Matthews Correlation Coefficient
DGA	: Domain Generation Algorithm
BDS	: Botnet Defense System
IOT	: Internet of Things
HTTP	: Hyper-Text Transfer Protocol
ACC	: Accuracy
Pre	: Precision
Rec	: Recall
F1	: F1 Score
NAT	: Network Address Translation
HD	: Hellinger Distance
CA	: Classification Accuracy
ROC Curve	: Receiver Operating Characteristic Curve

CHAPTER I

INTRODUCTION

1.1 STATEMENT OF PROBLEM

With the developing technology, many services are offered online. Online services have many advantages. These advantages can be generalized as ubiquitous access to services, reduced processing time, and elimination of physical intensity. Naturally, these services also increase the quality of service. One of the methods frequently used by attackers to interrupt these services is DOS (Denial of Service) attacks. In DOS attacks, the attacker aims to render the systems unusable by sending excessive requests and traffic to the systems. Such attacks are mostly made from a single source. In order to circumvent the security measures taken against these attacks, it is necessary to present the attack as if it is sending requests from different sources. These attacks are called DDOS (Distributed Denial of Service) attacks. In order to ensure that the attacks come from different sources, a botnet network called Zombie should be established. This way, commands are sent to other devices in the botnet network through a computer called the administrator, allowing millions of devices to send valid requests to a single resource. The way to stop the attack is to detect this attack, which is hidden in legal traffic, from different sources. Although botnet detection is not easy, it has been tried to prevent attacks with different approaches.

The main source accepted for DDOS detection is network traffic. On a serving server, the traffic volume is high. The attack time is unknown, as a normally running system can be attacked suddenly. Considering that the request to the server and the return time are also very low, there is not much time to examine. However, work is being done on a live system and false positive situations are not desired. As a result, it is necessary to prevent harmful packages from being hidden in normal traffic unexpectedly and not interfere with harmless ones.

Attackers also focus on weaknesses where they will have an advantage. The most advantageous situation for attackers is to hide inside normal traffic. A request

is sent to the server and its result is returned. The attacker is only interested in sending packets to the victim. So it can mask the sender's information. It can make the request through bot devices called zombies. It can even pretend to be a victim machine and send requests to external servers. Replies from millions of servers will be forwarded to the victim.

Two different approaches have emerged for botnet detection. Botnet detection systems such as signature-based, threshold-based, and ranking-based, which are traditional methods, and new generation, behavior-based botnet detection systems based on artificial intelligence.

There are different approaches to traditional methods. These approaches adopt the understanding of detecting and blocking an attack through known traffic.

Signature-based systems have a database of malicious traffic or requests. If there is a match between the traffic and the records in the malicious database, the botnet behavior is detected. In Threshold-based and ranking-based detection systems, if a request or traffic is received above a certain value, the incoming resource or request type is blocked. In traditional systems, before a malicious activity can be detected, it must be known beforehand. If values such as source address, message content, and source port number are easily manipulated, signature-based systems will not be very useful for botnet behavior detection.

Another important botnet detection system is the behavior-based system. Artificial intelligence plays an important role in behavior-based systems. Existing network traffic is examined and the behavior of the packets is tried to be analyzed. As a result of the analysis made with artificial intelligence, it is determined whether the traffic is harmful or not. The working artificial intelligence can take different measures according to the features it deems appropriate according to the current situation. This reduces the false-positive rate.

Within the framework of the above-mentioned, several main problems arise.

* The fact that DDOS attacks are important enough to cause service interruption is the main source of the problem.

*Due to the fact that DDOS attacks come from different sources over the network, difficulties in detection are another problem. Different attack techniques have been discovered by the attackers over time, and legal requests have been also used for attack purposes.

*A different problem is that the methods for detecting DDOS are insufficient. The traditional methods used prevent known attacks. DDOS detection systems have been developed with the help of artificial intelligence for new attack methods.

*The selection of the artificial intelligence algorithm to be used also plays an important role. There are many artificial intelligence algorithms and they all have different features. Correct algorithm and parameter selection will also positively affect the result.

*Artificial intelligence algorithms make predictions based on the features in the data. The network packages used in the study have many features and it is important to choose the most optimal feature.

CHAPTER II

REVIEW OF LITERATURE

2.1 BACKGROUND

The purpose of a DDOS (Distributed Denial of Service) attack is to interrupt services by overusing resources. System administrators' main goal is to maintain the availability of the services they deliver. In terms of continuity of access, measures are taken by backing up the energy systems, IT systems, and network devices used by the services. The resources used to access the systems can be specified as network access, web server, or other service. Each resource has a limited service capacity. When the demands above the limit come, the resources become unresponsive. Because attackers are aware of this vulnerability, they attack in different ways to render resources unresponsive. Thousands or even millions of devices are needed to carry out these attacks. For this, mobile phones, computers, or IoT devices are used. Malware is placed on the relevant devices and the control of the devices is taken. At the time of the attack, computers called zombies are activated by sending a remote command and sending a request to the victim machine. Thus, it is aimed to prevent or slow down access to services by sending requests far beyond the capacity of the systems. Such attacks are called DDOS (Distributed Denial of Service) attacks.

2.1.1 DDOS

Today, services are offered online. Online services have many advantages. Services can be accessed from anywhere, reducing processing time and eliminating physical congestion. These results also increase the quality. It is not desirable that the structure created to provide better service becomes unusable. Due to the importance of the situation, attackers send excessive requests and traffic to the systems, making the system unusable. While attackers want to prevent service access, security experts take the necessary precautions to ensure that the services continue uninterrupted. So much

so that sites such as GitHub and Twitter, which serve worldwide, are also exposed to DDOS attacks.

2.1.2 Types of DDOS

Volume-Based DDOS (Volume-Based Attack): It is a system that provides intensive queries to instantly fill the bandwidth service used on the servers. It is the most used DDOS attack model worldwide.

Protocol-Based DDOS (Protocol-Oriented Attacks): There are various layers within the Open Systems Interconnection (OSI). This is done by using the vulnerabilities in units 3 and 4 within the layers. It is an attack model that is dangerous and locks systems (server, db, etc.).

Application Layer DDOS (Application Layered Attacks): These are the attack units that create load on the server by using the forms with GET and POST features in the systems hosted in the server content.

SYN Flood DDOS: On the server side, TCP-focused resource packets can pose a serious threat. These bundled files are the most serious problems in server systems and make source data unusable. Therefore, it draws attention as one of the most dangerous attack models.

UDP Flood DDOS: These are the attack types used to lock the ports running on the server side. It is a DDOS model that allows ports to be closed or unable to serve by sending UDP packets.

Ping Flood: As the name suggests, it is an attack model that occurs as a result of “PING” in the server wing over thousands or even millions of IPs.

Attack classifications ;

- Network Level Attacks: The most basic - TCP, UDP, ICMP, Floods
- Reflective / Amplified Attacks: Service-oriented ones - DNS, NTP, SNMP, SSDP
- Fragmentation: Session specific
- Application Specific: Repeated GET, slow READ, or loop calls
- Crafted: Stack and protocol level, buffer resources

2.1.3 Network Packets

The communication of two or more devices with each other is called a network. The protocols of the devices are standardized with the OSI (Open System Interconnection) architecture.

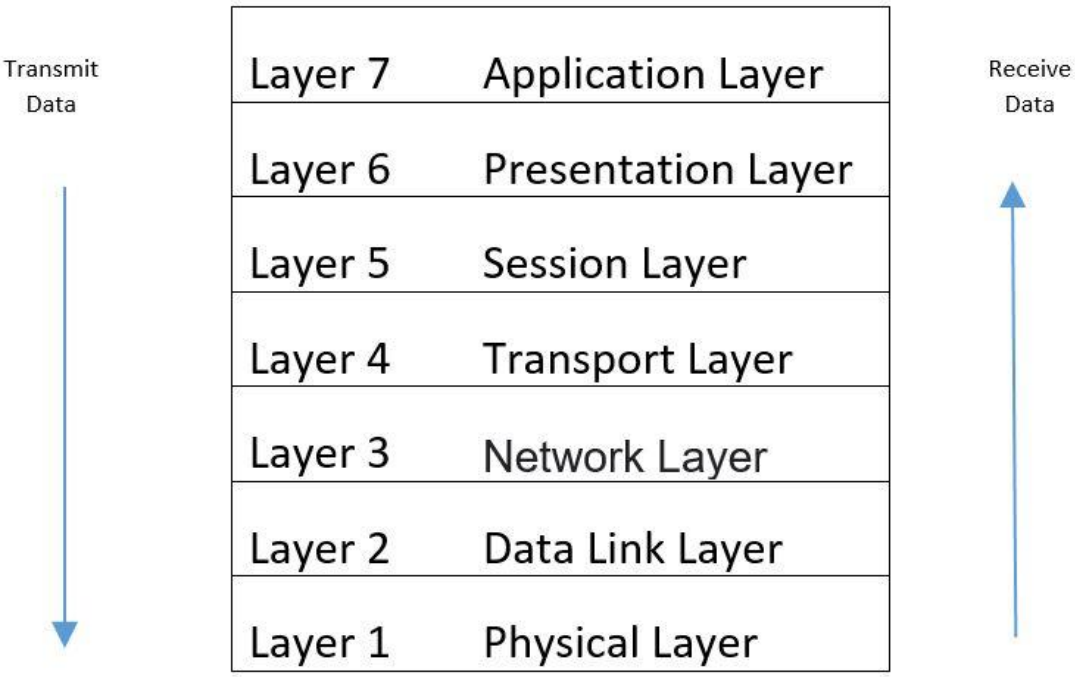

Figure 1: OSI Model

As shown in Figure 1, OSI consists of 7 layers. When sending, data is transmitted from top to bottom. Each layer adds its features and transmits data to a lower layer. Finally, it is transmitted from the physical layer to the cable. The features to be examined are in the IP layer. Physical Layer: It is the hardware layer and the first layer. In this layer, it is ensured that the data is transmitted over the cable as bits (0 and 1). While it converts 1 and 0 to electrical signals on the sending side, it converts these signals coming from the cable back to 1 and 0 on the receiving side.

Data Link Layer: It is the layer related to accessing and using the physical layer.

Network Layer: It is the layer where the IP structure shown in Figure 2 is used and the inter-network access rules are included.

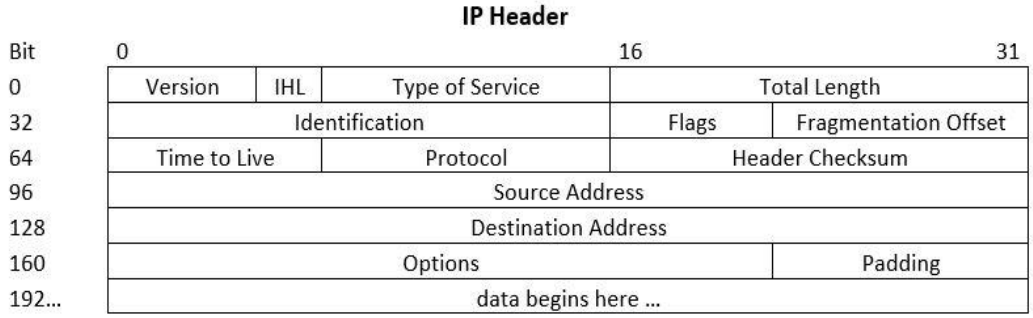

Figure 2: Ip Header

Transport Layer: It segments the data from the upper layer and sends it to the lower layer. It also combines the data from the lower layer and transmits it to the upper layer. TCP and UDP protocols work at this layer. TCP and UDP packet structures are shown in Figure 3, each packet has its unique fields.

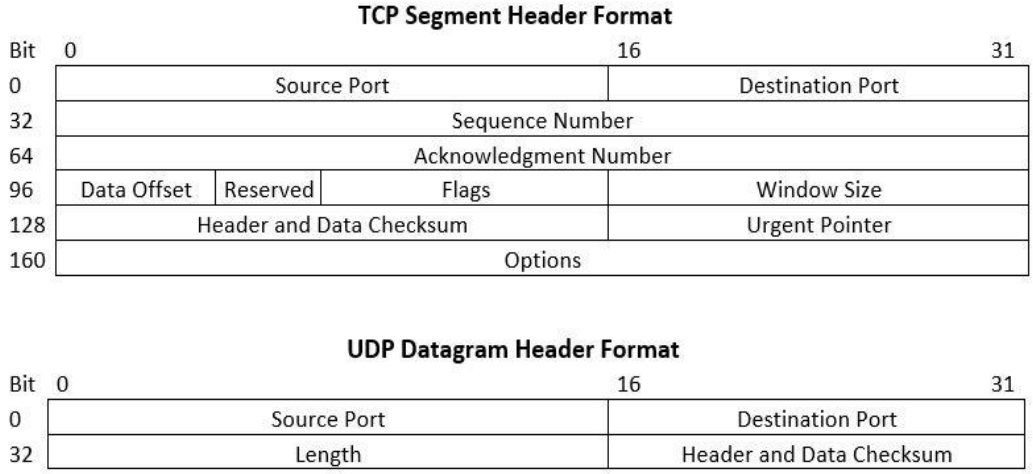

Figure 3: UDP – TCP Header Format

Session Layer: It is the layer where the connection between applications is established, managed, and terminated. When a computer is communicating with multiple computers at the same time, it ensures that it can talk to the right computer when needed.

Presentation Layer: It allows the sent data to be understood by other computers. The data is sent to the application layer, and the structure of the data is edited. The format of the data is determined in this layer.

Application Layer: The application layer is the layer closest to the end user. It receives information directly from the user and transmits the incoming data to the

user. In this layer, the requirements of the users are met. Applications are enabled to run on the network.

2.1.4 BOTNET

DOS (Denial of Service) attacks are made by sending excessive traffic or requests from a single source to the victim machine. The attack can be stopped by interrupting the access of the source device to the system.

By attacking many sources, it has been tried to make it difficult to detect and prevent the attack. For this situation, different devices to be used as attack sources are needed. Devices surfing the Internet have been infected with malicious codes without their knowledge and have been taken under control. At the time of the attack, they used it to generate malicious traffic. These devices have been called zombies or the zombie botnet community. In short, a botnet is a collection of devices that are managed from a control center and take action on demand.

Since botnet is a type of attack that appeared many years ago, In [1] , the botnet lifecycle has been given in Figure 4.

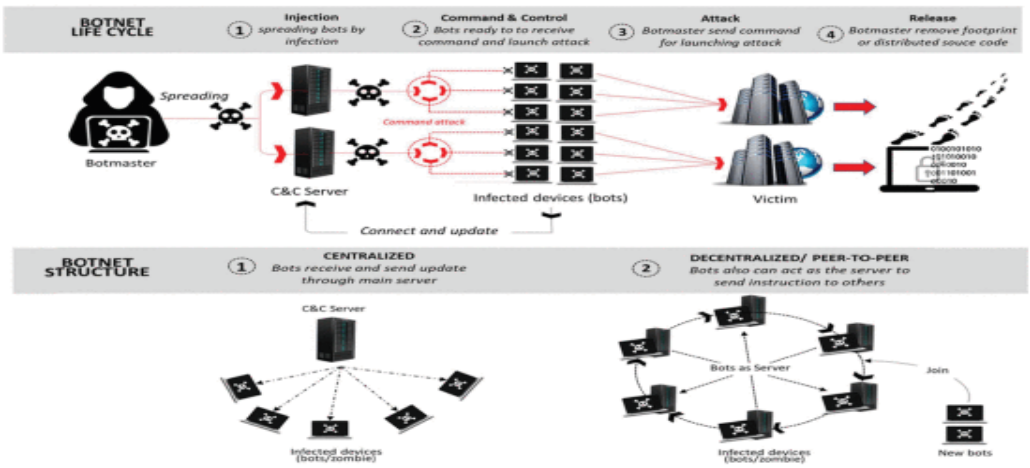

Figure 4: Life Cycle of Botnet [1]

In Figure 4, The first part is the infection stage. In this stage, the botmaster (attacker) tries to spread the bot by infecting a system. It spreads through methods such as downloading files over the Internet and sharing via e-mail, the web, or social media. At this stage, the goal is to increase the number of bots. First, the bots will be

downloaded and installed, then the device will become a botnet element and can be managed remotely.

The second stage is the Command and Control (C&C) phase. The botnet sends a report to its administrator (Bootmaster). Botmaster sends the necessary codes and commands to the botnet after this report. The botnet does not contain any malicious code in itself, there is no communication between them during the waiting phase, and the botnet waits silently. For these reasons, the botnet is not detected by security devices.

The third stage is the attack stage. The number of bots must be at the desired level for this stage. Commands are sent to each bot by the bot manager and they are all made to attack the same target. An example of these attacks was the attack on GitHub in 2018 (1.35 Tbps) interrupted for 10 minutes. The other notable DDOS attack was the Mirai attack in 2016. In this attack, hundreds of web pages, such as Twitter, Netflix, Reddit, and GitHub, have been affected for a few hours by the packets sent by 400,000 IoT devices.

The final stage is the Release stage. At this stage, the bots are released, and the actions to be done with the bots are finished. Some botmasters publish codes related to bots and remove their footprints. The command and control phase is the greatest moment to find the botnet. It is difficult to detect at the infection stage because there are many forms of transmission. In the attack phase, it will be late.

Two different structures can be mentioned for botnets.

The first stage is centralized: The C&C server is centrally located in this section. Receive and send update packages via the C&C server.

The second stage is the Decentralized/Per-To-Peer stage: Different botnets act as servers in this section. Messaging is provided between them via botnets that act as servers.

2.1.5 Algorithms and Data Structures

2.1.5.1 Machine Learning (ML)

It is the process of learning by giving data beforehand in order to enable the computer to learn like the human mind. As a result of this learning, what is expected is the correct prediction of the computer for similar events.

At this learning stage, it is necessary to have appropriately formatted data sets. Datasets should contain features of the event to be learned.

The data is divided into categorical and numerical. Numerical data indicate quantity and are continuous, while categorical data are data expressed by classification, they indicate quality.

Categorical data such as status and direction are used in classification, and numerical data such as port number and length are used in estimation algorithms.

Supervised Learning: The training data for the supervised learning approach includes "label" information. In order to construct a model for the solution, data with known results are used. In this way, it is aimed to predict the results of the data without label information in the data set, based on the model created.

Unsupervised Learning: There is no label information in the unsupervised learning process. Based on the elements in the data set, it aims to reveal hidden relationships or groups.

Overfitting: The main purpose of machine learning applications is to obtain patterns from the data at hand and to make accurate predictions by using these patterns for new data. In the model, the risk of overfitting is high if more than necessary memorization is made using the training sets or if the training set is monotonous. As a result of overfitting, memorization may occur, not learning.

2.1.5.1.1 Decision Tree

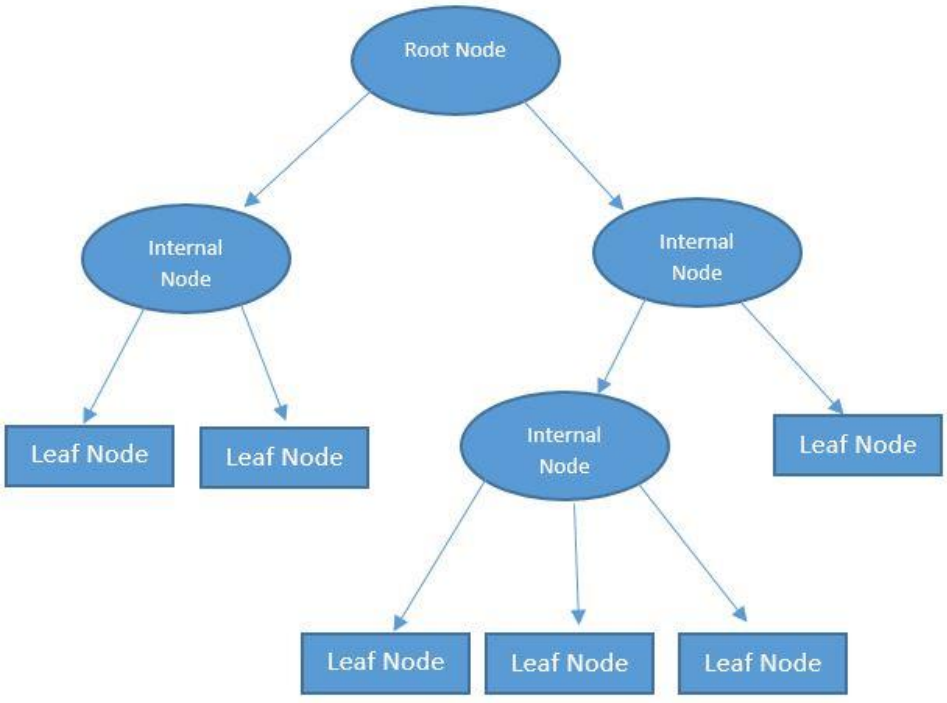

Figure 5: Decision Tree Structure

One of the data mining classification algorithms is the Decision Tree algorithm. As shown in Figure 5, a Decision Tree is a structure used to apply a set of decision rules to break up a dataset with many records into smaller sets. In other words, it is a framework that divides enormous volumes of records into little record groups by the application of basic decision-making procedures. They can be adapted to the solution of all problems (classification and regression).

Advantages of Decision Tree:

- It is simple to understand and interpret. Decision tree models can be understood with a simple explanation.
- Experts' descriptions of a situation (alternatives, possibilities, and costs) and outcome preferences can be used to make significant forecasts.
- Finding the worst, best, and anticipated values for various scenarios is helpful.
- It can be combined with other decision techniques.

Disadvantages of Decision Tree:

- They are unstable, which means that a slight change in the data might result in a big change in the optimal Decision Tree's structure.
- They frequently err. With comparable data, several different prediction methods perform better. A Random Forest can be used to solve this problem instead of a single decision tree, however it is more difficult to read than a single Decision Tree.
- Calculations can become quite complicated, especially if numerous numbers are ambiguous or highly associated with various outcomes.

2.1.5.1.2 Naive Bayes

The Bayes theorem is the foundation of the Naive Bayes classifier. Although it is a sluggish learning approach, it can still be used with uncertain datasets. The algorithm sorts elements according to the highest probability value after computing the probability of each state for each element. With little training data, it can produce extremely effective works.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

P(A|B) = Conditional probability of A given B.

P(B|A) = Conditional probability of B given A.

P(A) = Probability of event A.

P(B) = Probability of event B

Advantages of Naive Bayes:

- It performs better than models like Logistic Regression since each feature is viewed as being independent of the others.
- Simple and easy to use.
- Small data sets produce successful outcomes.
- Both continuous and discrete data can be used with it.
- It can also be applied to data sets with imbalances.
- It can function effectively in high dimension data.
- Due to its speed, it can be employed in real-time systems.
- It doesn't engage with superfluous features.

Disadvantages of Naive Bayes:

- Every aspect in real life has some degree of interdependence.
- Because operations are carried out under the assumption that the attributes are independent of one another, relationships between variables cannot be described.
- Problems with Zero Probability could occur. When the intended sample is absent from the data set, there is zero probability. Consequently, it will produce a result of 0 when used in any operation. The most straightforward solution is to completely rule out this option by adding a minimal value (often 1) to every data.

2.1.5.1.3 K-Nearest Neighbor (K-NN)

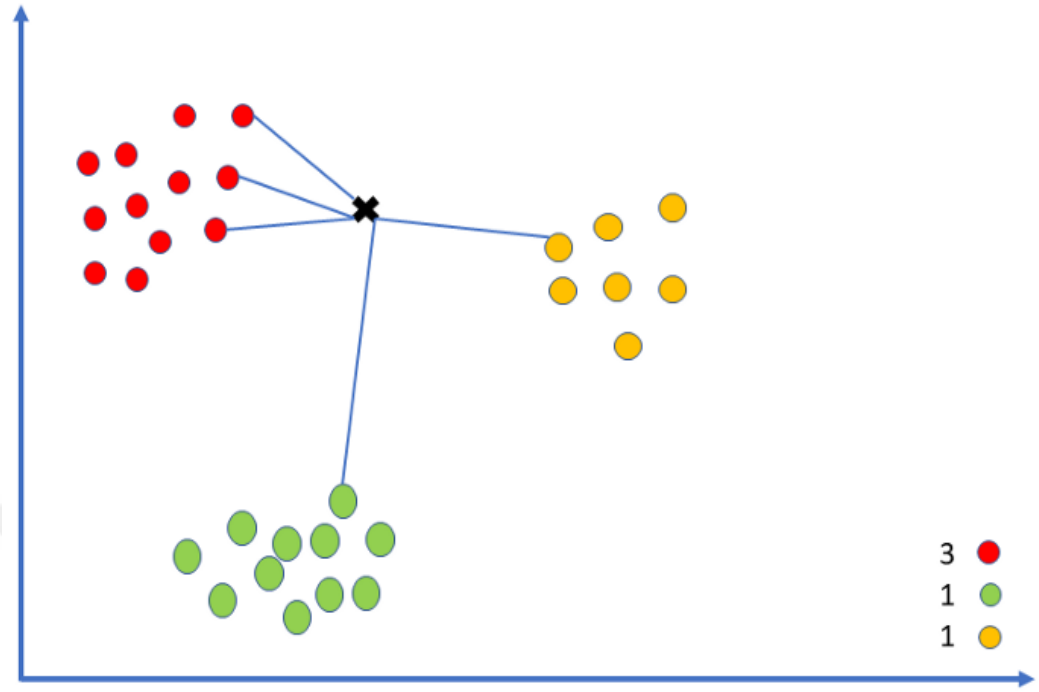

Figure 6: K-NN Algorithm [43]

The K-nearest neighbor (K-NN) algorithm is one of the supervised learning algorithms that are easy to implement. Although it can be used to solve classification and regression issues, the majority of industrial applications are for classification issues. T. M. Cover and P. E. Hart proposed K-NN algorithms in 1967. By utilizing the data in a sample set with particular classes, the algorithm is applied. As shown in Figure 6, the distance between the new data and the current data that will be included to the sample data set is calculated, and k number of close neighbors are examined. Due to its resilience to outdated, straightforward, and noisy training data, K-NN is one of the most often used machine learning methods. However, it also has a drawback. For instance, it uses a lot of memory when working with big amounts of data because it maintains every state when computing distances.

How to work

- The parameter k is established first. The number of closest neighbors to a given place is this parameter.
- The two closest neighbors will be used to classify data if k=2.
- Using the distance function and the available data, the distance of each data point in the data set is determined.

- The k nearest neighbors are identified using the calculated distances. According on the attribute values, it is assigned to the class of neighbors.
- The chosen class is regarded as the anticipated class. The new data has a class assigned to it.

Advantages of KNN:

- The training process is typically simpler than those of other algorithms,
- Process and analysis traceability using analytical/numerical methods is offered.
- Effective when training with complex or noisy data,
- Easy to modify.

Disadvantages of KNN:

- Due to the large transaction volume and transaction step, it demands expensive hardware.
- Due to the numerous processes and transactions, it takes time even if it is resistant to high-volume data.

  Finding the right algorithm for performance (Distance equation, parameters, etc.) can occasionally take a while.

2.1.5.1.4 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning method based on statistical learning theory. In essence, it forms a line to categorize points that are placed on a plane. This line is intended to be as close to both kinds of points as possible. Makes this separation according to the elements at the boundary. It works well with complicated but modest to medium-sized datasets. It is one of the very effective and simple methods used in classification.

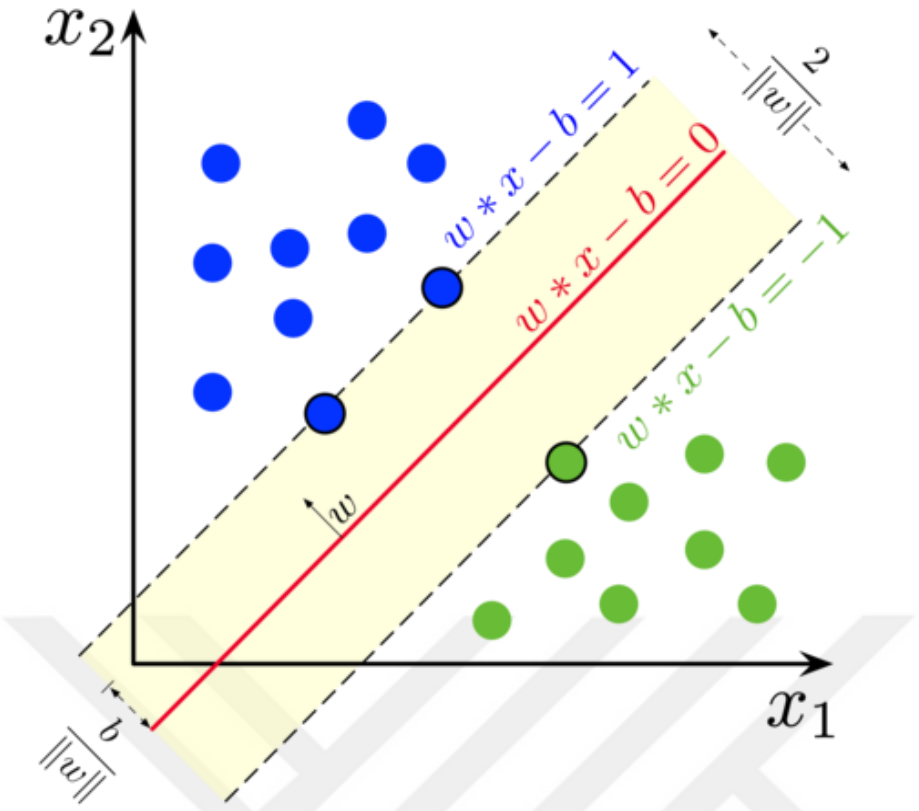

Figure 7: SVM Algorithm [44]

There are two different classes in Figure 7, blue and green. The main purpose of classification problems is to decide in which class the future data will take place. The green area between the line that divides the two groups used to make this classification is known as the Margin. The separation of two or more classes is improved by a broader margin.

The following formula is used in decision-making.

$$\hat{y} = \{-1 \; if \; w^T.x + b < 0, +1 \; if \; w^T.x + b \geq 0, \qquad (2.2)$$

w; weight vector (θ1), x; input vector, b; is the deviation (θ0).

If the result for a new value is less than 0, it will be closer to the green dots. If the result is greater than or equal to 0, then it will be closer to the blue dots.

Hard Margin And Soft Margin

The margin may not always be as seen in Figure 8 below. Sometimes values can stay in the margin area and this is called Soft Margin. If the data is separated

linearly, this situation is called as Hard Margin. Hard Margin is very sensitive to outliers. Therefore, Soft Margin can be preferred.

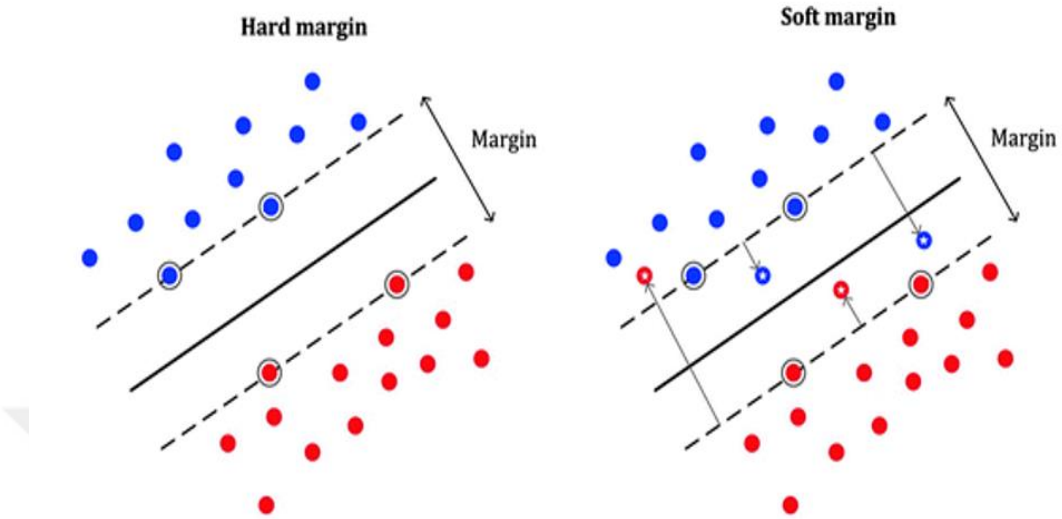

Figure 8: SVM Soft Margine Algorithm [45]

The balance between the two can be checked with the C hyperparameter in the SVM. As seen in Figure 9, the larger the C, the narrower the Margin.

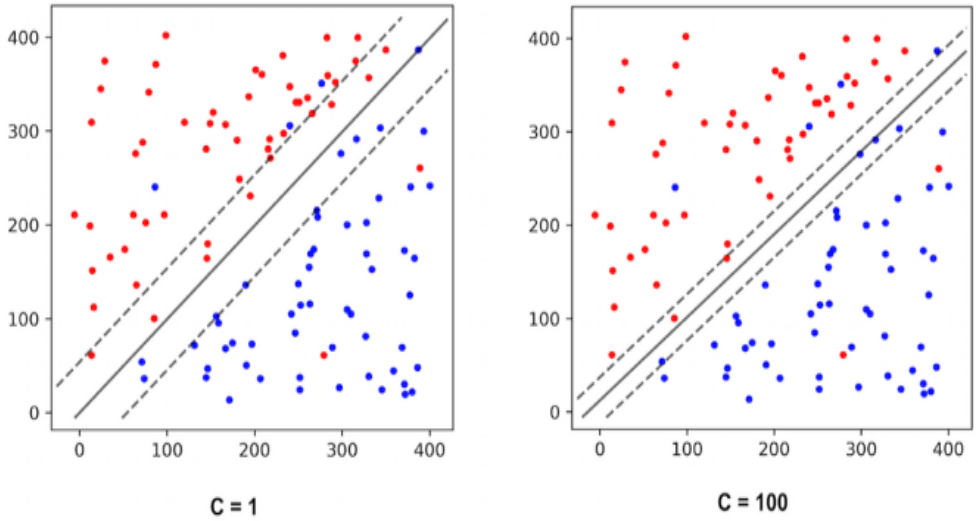

Figure 9: SVM Overfitting Algorithm [46]

Also, if the model is overfitting, C needs to reduce.

2.1.5.1.5 Random Forest

One of the supervised classification techniques is the Random Forest algorithm. It is applied to both classification and regression issues.

As shown in Figure 10, it creates a forest by generating multiple decision trees and does it somehow randomly. Its purpose is to increase the classification value during the classification process. The decision trees in the "forest" he created were all trained using the "bagging" technique. The bagging method's general tenet is that combining learning styles improves final results.

While generating trees, Random Forest increases the model's unpredictability. When splitting a node, instead of looking for the most important feature, among a random subset of features, it looks for the best feature. This leads to a large variation, which frequently produces a superior model. Therefore, the process for dividing a node in Random Forest only takes into account a random subset of features. Trees can be made more random by utilizing random thresholds for each characteristic rather than searching for the optimal thresholds (like a typical decision tree).

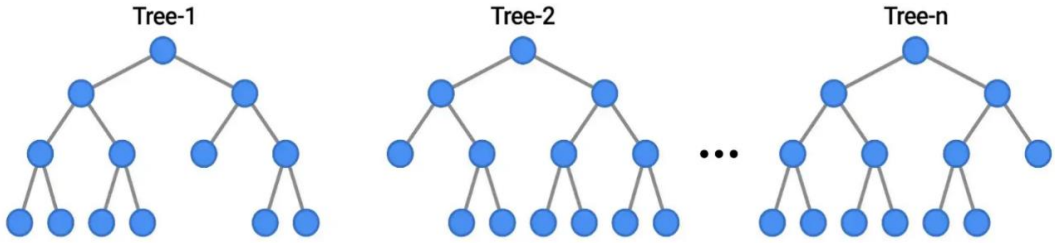

Figure 10: Random Forest Algorithm

2.1.6 Machine Learning Tools and Algorithms

Machine Learning Tool: As it is known, the most suitable software language for Machine Learning algorithms is Python. There is software that automates with the help of the interface without writing code, as well as the operations, can be done on code with Python. The "Orange" program has been used in the study.

Orange is a machine learning visualization tool that is free and available to both novices and experts. It is an interactive data analysis tool that has a large toolbox and enables workflows with widgets.

The parameters used in the algorithms work in default settings and can be changed in the software. Below are the algorithms that can be changed in Orange software.

2.1.6.1 Random Forest Parameter

Number of trees: Decide how many decision trees the forest will contain.

Number of attributes considered at each split: Sets the number of qualities that are chosen at random and assessed at each node. The default value is the square root of the total number of data attributes.

Replicable training: Ensures repeatability of outcomes by updating the seed for tree growing.

Balance class distribution: The relationship between weight classes and their frequencies is inverse.

Limit depth of individual trees: It is possible to alter the depth at which the trees will grow.

Do not split subsets smaller than: Picks the smallest splittable subgroup.

2.1.6.2 KNN Parameters

Number of nearest neighbors: Weights and the distance parameter (metric) are model criteria.

Metric:

Euclidean: “straight line”, the separation between two points

Manhattan: Total absolute disparities across all characteristics

Maximal: Absolute disparities between qualities that are most significant

Mahalanobis: The separation between a point and its distribution.

Weights:

Uniform: Every point in every neighborhood has the same weight.

Distance: A query point's immediate neighbors have more influence than its distant neighbors.

2.1.6.3 Decision Tree Parameters

Induce binary tree: construct a binary tree (two child nodes are created).

Min. number of instances in leaves: The decision of whether to separate each branch

Do not split subsets smaller than: Prevents the algorithm from dividing the nodes that have less instances than the specified number.

Limit the maximal tree depth: The number of node levels supplied should reflect the maximum depth of the classification tree.

Stop when majority reaches [%]: Once a certain majority criterion has been met, stop splitting the nodes.

2.1.6.4 SVM Parameters

SVM type: SVM and v-SVM are based on various error function minimizations. Limits on test errors can be imposed.

SVM Cost: Loss-related word that relates to classification and regression problems.

SVN ε: Regression problems are covered by a model parameter for the epsilon-SVR. Specifies the range of genuine values within which forecasted values are not subject to a penalty.

ν-SVM Cost: Loss-related word that only applies to regression tasks.

ν-SVM ν: A The v-SVR model has a parameter that affects both classification and regression tasks. a lower bound on the percentage of support vectors and an upper bound on the percentage of training mistakes.

The approach can build the model with linear, polynomial, RBF, and sigmoid kernels since the kernel is a function that converts attribute space to a new feature space to suit the maximum-margin hyperplane. When a function is chosen, the kernel is specified, and the relevant constants are:

g : Since there may not be a training set provided to the widget, the default value for the gamma constant in the kernel function is 0 and the user must manually modify this option, even though the suggested value is 1/k, where k is the number of attributes,

c : Relating to the kernel function's constant c0 (default 0)

d : With relation to the kernel's degree

Numerical Tolerance: Sets the Numerical Tolerance's allowable variation from the target value.

Iteration Limit: Checking the box Setting an iteration limit will determine how many iterations are allowed.

2.1.6.5 Machine Learning Metrics

AUC (Area under the ROC Curve): AUC offers a comprehensive evaluation of performance across all available classification factors. One way to analyze AUC is to consider the likelihood that the model values a randomly positive sample higher than a randomly negative one.

CA (Classification Accuracy): Accuracy is one factor to consider when rating categorization models. Accuracy is the proportion of forecasts that our model successfully predicted. The official definition of accuracy is as follows:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{2.3}$$

F1 Score: A machine learning evaluation metric called the F1 score assesses a model's accuracy. It combines a model's precision and recall ratings. How many times a model is correctly predicted throughout the full dataset is determined by the accuracy statistic. This measure can only be trusted if the dataset is class-balanced, meaning that each class contains an equal amount of samples.

Precision: Precision, or the caliber of a successful prediction made by the model, is one measure of the model's performance. Precision is calculated by dividing the total number of positive predictions (the sum of the true positives and false positives) by the number of true positives.

Recall: The recall is determined as the proportion of Positive samples that were correctly identified as Positive to all Positive samples. The recall gauges how well the model can identify positive samples. The more positive samples that are identified, the larger the recall.

2.2 LITERATURE

DDoS detection is the process of separating DDoS attacks from normal network traffic to achieve effective attack mitigation. Identifying their characteristic features is the first step in detecting DOS/DDOS attacks. Up to now, various research has focused on the technical and behavioral analysis of DDoS attacks.

Gaurav provides information on the definition and types of DDoS attacks, followed by examining the effects and damages caused by such attacks. Subsequently, various methods and approaches used for detecting DDoS attacks are discussed. The advantages, disadvantages, and application areas of each approach are thoroughly examined in the article using an academic perspective [2].

In this study, the types of DDOS attacks, their effects on the network, and measures to protect against these attacks are investigated. Details of attacks are described in 3 main environments.

- Attacking Environment: 2 basic components make up the attack part, Bot, and Botmaster.

- Handling Environment: There are 3 handler components as C&C server, Report server, and Loader.

- Target Environment: It consists of 2 components: The newly infected device and Victim [3].

In 2012, Alomari et al. focused on the risk of DDoS attacks in services and services running on servers. Different attack models and botnet-based DDoS attacks are discussed in the article. They classified botnets according to DDoS attacks [4]].

In the following sections, botnet-based DDoS attacks will be focused on and these will be examined in two groups; traditional methods and AI-based methods.

2.2.1 Detection Using Traditional Methods

Various methods have been developed for detecting DDOS attacks. The initial methods developed for attacks are called traditional methods, which include packet filtering, network traffic analysis, examining the effects of attacks, and signature-based detection systems.

2.2.1.1 Signature-Based Detection

The signature-based detection system is a traditional method that is used for known pests. These attacks are designed to detect known attacks using signatures and effectively detect known attack types without generating false alarms. Buczak et al. stated that the database had to be manually updated with rules and signatures and could not detect new (zero-day) attacks [5].

The article discusses the importance, types, and effects of DDoS attacks. Then, the limitations of signature-based methods and the use of machine learning

techniques to overcome these limitations are discussed. The proposed method focuses on creating traffic signatures and using machine learning algorithms to detect attacks. The technique uses a multi-stage process to reach the result.

It is also noted that the proposed method can be further developed to increase its effectiveness in the case of a real-time DDoS attack. However, it is mentioned that more data collection and the creation of more classification models are needed to improve the method's accuracy.

In the conclusion section, it is emphasized that the proposed method is an effective alternative that can be used to overcome the limitations of signature-based methods [6] .

The causes, types, and effects of DDOS attacks, as well as the use, advantages, and disadvantages of signature-based methods, are examined in this paper. The working principle of signature-based methods is discussed in detail. These methods analyze network traffic using pre-defined signatures and detect attacks by identifying matching signatures. The paper explains how these methods work and their advantages and disadvantages.

In addition, examples of different signature-based methods are provided in the paper. For instance, open-source security tools like Snort use signature-based methods to detect attacks. These tools detect attacks by using pre-defined signatures and prevent attack traffic.

The paper demonstrates that signature-based methods are widely used for detecting and defending against DDOS attacks. However, the disadvantages and limitations of these methods are also stated in the paper [7].

2.2.1.2 Threshold-based Detection Methods

By modeling normal network behavior, it is detected as an attack when a certain criterion in traffic (usually traffic density) exceeds a specified threshold. However, having a sufficient amount of normal traffic data is imperative, and behavioral changes need to be noted.

The study conducted a series of experiments to identify effective features and determine their threshold values for detecting different types of DDoS attacks.

Several experiments investigated the effects of different features and threshold values. As a result, certain features and threshold values are more effective in detecting various types of DDoS attacks. However, the authors have emphasized

that relying solely on threshold-based methods has some disadvantages, and feature-based methods should also be used.

The study demonstrates that traffic features and threshold-based methods effectively detect and classify DDoS attacks. However, using multiple methods together for detecting and classifying different types of attacks is recommended [8].

The article compares two DDoS attack detection methods, Threshold-based and Anomaly-based, in software-defined networking (SDN) environments. Firstly, common characteristics of DDoS attacks are identified, and then measurements that can be used to detect these characteristics are examined.

In addition, an SDN testbed is created using the OpenFlow protocol and Ryu controller, and DDoS attacks are detected through simulations performed on this testbed.

As a result of the simulations, it is determined that the Anomaly-based detection method is more successful than the Threshold-based method. However, it is concluded that both methods can be used together, increasing the detection rates of DDoS attacks even further [9].

2.2.1.3 Ranking-Based Detection Methods

A ranking-based method is proposed for detecting and preventing DDoS attacks. The main idea of the study is to evaluate the features that distinguish attacker traffic from regular traffic based on a ranking system.

A test environment is created to collect attacker and normal traffic data and calculate ranking-based features. Then, the values of these features are gathered in a ranking table, and a threshold value is determined. All features above the threshold value are defined as attacker traffic and considered as a DDoS attack.

In addition, the effectiveness of the proposed method is tested in several scenarios. The tests reveal that the proposed method has higher accuracy rates and is more successful in detecting attacks than other methods.

The study proves that the proposed ranking-based method is effective for detecting and preventing DDoS attacks in SDN environments [10].

A ranking method is proposed for detecting DDoS attacks in an SDN environment.

Using a ranking method determines the differences between attacker and regular traffic. For this purpose, the collected traffic data is analyzed, and a feature

vector is created. This feature vector determines the differences between attacker and regular traffic. Then, the features in the feature vector are ranked, and a weight value is assigned to each feature. These weight values are used to determine the importance level of each feature.

Next, a ranking table is created, and a threshold value is set for each feature. These threshold values determine the differences between attacker and regular traffic. All features above the threshold value are identified as attacker traffic and detected as DDoS attacks.

The study demonstrates the usability of a ranking-based method for detecting DDoS attacks in an SDN environment. However, more work and testing are needed to determine the method's effectiveness in real-world scenarios [11].

2.2.1.4 Other Traditional Detection Methods

Botnet detection has been examined over a specific pattern, and traffic pattern analysis has been used. The study aims to identify malicious activities by formulating properties in network traffic. The system, which has been also tested with actual data, can detect malicious activity in a short time [12].

Zhao et al. have worked to catch botnets at the command and control stage. In the study, it has been tested on network packages with artificial intelligence models, and a success rate of over 90% has been achieved [13].

2.2.2 Artificial Intelligence-Based Detection:

Signature-based systems have become ineffective against unknown attack methods since they can only detect known attacks. Research on artificial intelligence has been increased in DDOS detection and prevention against the problems of method, source device, ports used, C&C communication methods, and continuous differentiation in DDOS attacks.

Hall et al. While explaining the necessity of artificial intelligence, it is stated that if the event is defined as past and future, training data from the past will be learned, and these data will be used as test data in the future prediction phase [14].

Attackers make attacks with different characteristics from different sources each time. More dynamic Machine Learning method, threats can be detected much faster and easier than the traditional method [15]

Discusses a machine learning-based system proposal for detecting DDoS attacks in cloud environments. While providing detailed information on DDoS attacks and their potential impact, particularly in cloud environments, different methods for detecting them are also discussed.

The proposed system is designed to detect DDoS attacks using machine learning algorithms. The system processes network traffic data using feature extraction methods and is trained with machine learning algorithms. Subsequently, simulations are conducted in different scenarios to test the system's performance.

The simulations conducted with the proposed system demonstrate higher accuracy rates than other methods. Furthermore, the system is specifically designed to be used in cloud environments, considering scalability and ease of use [16].

2.2.2.1 Dataset and Feature Extraction in AI-Based Botnet Detection

The article discusses the development of a realistic attack dataset and taxonomy for distributed denial of service (DDoS) attacks. The challenges encountered in the process of creating a realistic DDoS attack dataset and taxonomy, as well as how to overcome these challenges, are also addressed. It is also noted that detecting DDoS attacks is complex, and the reasons for the difficulty include IP spoofing, IP redirection, MAC address leaking, and modern attack tools [17].

The most crucial point for DDOS attack detection is whether the features required for classification are selected correctly. Therefore, effective feature selection plays an important role in making an efficient DDoS detector [18].

It is stated that attack types differ, and it is necessary to analyze different parameters to define different attack types. Although the source ip, destination ip, and port number are similar in an attack type, these parameters will vary in various attacks. There needs to be more than these for real-time attack detection. For DDOS detection, other parameters need to be analyzed as well [19].

2.2.2.2 Anomaly Detection

A study has been conducted by Aburomman et al. to investigate the effectiveness of combining multiple classifiers to achieve better results, as the authors have held the belief that a strong classifier would lead to a favorable model. This combination states that classifiers will form a community using the methods of Bagging, Boosting, Stacking, and Mixtures of Competing Experts. The study

emphasizes the importance of the reliability of classifiers. The accuracy may be further improved if the reliability rate is accurately estimated in the voting among classifiers for the decision-making process [20].

A recurrent deep neural network has been designed by Yuan et al. and compared with artificial intelligence. The bidirectional recurrent neural network model and Random Forest algorithm have been studied, and they state that they will reduce the error rate from 7.517% to 2.103% [21].

Das et al. Network Intrusion Detection System (NIDS) has been proposed to detect DDOS attacks. The feature of this system is the establishment of a decision mechanism in the form of a community. The data in the data set is tested with 4 different models (K Nearest Neighbor, Multilayer Perceptron, Support Vector Machine, J48), and it is decided whether it is harmful or not as a result of voting between them. When the effects are examined, it is shown that the ensemble classification model is better than any single class in terms of accuracy, TPR, and FPR [22].

Performance analysis of the most used machine learning algorithms by Tuan et al. Notable in the article is that it will address the scalability and robustness issues in previous research. Considering this problem, a performance analysis has been made. On two separate databases, analysis has been conducted using the Support Vector Machine (SVM), Artificial Neural Network (ANN), Naive Bayes (NB), Decision Tree (DT), and Unsupervised Learning (USML) models. The authors state that USML will give the best results for Accuracy, False Alarm Rate (FAR), Sensitivity, Specificity, False positive rate (FPR), AUC, and MCC values [23].

A 2-level architecture is recommended to detect and prevent botnet attacks. First, a deep learning model has been applied to understand the scanning efficiency in the early attack phase. In the second stage, a different deep-learning model has been used to detect DDOS attacks from the seized devices. Each stage of the model consists of data collection, data preview, feature selection, and scan detection stages.

The proposed approach has been applied in 4 different scenarios, 2 of them got results above 99%, and the other two got results above 97%. These results show that it will detect DDOS attacks efficiently.

The application is mostly designed for large-sized matrices such as image processing [24].

The method of preventing botnet attacks by capturing the communication between C&C and botnets has been tried.

The study aims to control DNS request domains. Domain Generation Algorithm (DGA) modeling uses different machine learning algorithms. In the study, the lengths of the domain names have also been considered, and the authors determine whether they will be legal or not by using Machine Learning algorithms. At the end of the study, the best result in detecting malicious domain names has been obtained with the SVM (Support Vector Machine Algorithm) model [25].

2.2.2.3 Other Research Using AI

In the article, two complementary models, Mathematical Model, and Machine Learning Model, are proposed to detect DDOS attacks. A mathematical model is proposed to predict the quantitative behavior of the system. Quantitative findings from mathematical models are compared with observational data. In the machine learning model, ML algorithms such as Logistic Regression and Naive Bayes are used to validate the proposed model. As a result, the logistic regression algorithm and the Naive Bayes algorithm have been compared and the accuracy of the Logistic Regression algorithm has been found to be better [26].

In the study with artificial neural networks, feature selection is emphasized. It has divided the features into 4 groups: Byte-Based Feature, Behaviour-Based Feature, Packet-Based Feature, and Time-Based Feature. The feature selection algorithm proposed by the authors has been used in the study. At the end of the study, the authors conclude that the time, byte, and behavior-based feature is efficient. They also state that the packet-based feature will not provide detailed information on packet exchange [27].

2.2.2.4 Botnet Defense System

A cyber security strategy called Botnet Defense System (BDS) employs white hat botnets to take out hostile botnets. White hat bots use up system resources while protecting the IOT system from malicious bots.

The approach consists of 3 stages.

1- Observability and Controllability: A graphic model that defines and communicates the interaction between BDS and IOT systems is used.

2- Monitoring: Monitoring the surroundings and finding harmful bots are stages of BDS.

3- Basic Command and Control Strategy: The BDS system manages two distinct white hat botnets that are uncontrollable. However, there is a direct connection between the two forms of botnets. All white hat botnets interact with BDS as a result.

The purpose of the pull-out strategy is to neutralize malicious botnets and delete any remaining white-hat botnets in the system [28].

In order to understand IOT botnets and botnet families, bot behaviors have been tried to be defined using the Extreme Learning Machine method.

The framework process consists of 2 steps. The first step examines the network flow and reveals the connection status. It also extracts the transition matrix from the connection states. A transition matrix is extracted for each state, and a final dataset is created by combining the state matrices. In the second step, a learning-based technique is used to discover bots and families.

Markov Chains have been applied to explore the botnet family behavior pattern. At the end of the study, the authors state that the accuracy and recall in botnet detection are above 97% [29].

2.2.3 DDOS Protection Methods

Different defense techniques used against DDoS attacks are examined, and information is provided about the advantages and disadvantages of each technique.

Comprehensive structures have been established to detect, mitigate and prevent DDOS attacks. The most appropriate and effective defense features, measures, and methods are discussed in the contents of these structures. This article provides a key point for developing advanced defense methods and solution models against DDoS attacks [30].

The article's main focus is techniques for detecting and preventing DDoS attacks. To this end, the article discusses network-based, server-based, and application-based detection and prevention techniques.

The defense mechanisms that can be used against DDoS attacks are examined in detail and divided into four main categories: network-based defense, server-based defense, application-based defense, and cloud-based defense. Each category discusses relevant defense mechanisms and their advantages and disadvantages in detail.

Moreover, the article highlights the importance of an interdisciplinary approach in preventing DDoS attacks and creating more effective defense mechanisms against them. The authors emphasize that an interdisciplinary approach can provide more comprehensive and effective DDoS defense solutions [31].

2.2.4 Artificial Intelligence Results in DDOS Detection

In another article using CIC-AWS-2018 and ISOT HTTP Botnet dataset, the success rate of Random Forest and Decision Tree models is higher than other machine learning models. Conversely, Naive Bayes is the most unsuccessful model. Therefore, the authors have seen that it would be possible to detect botnets with machine learning. In addition, they have determined that the Support Vector Machine would use the highest rate of resources compared to the others [32].

As a result of a different article, the predictive value is derived using the accuracy of the values (Acc), precision (Pre), recall (Rec), and F1-score (F1). Different algorithms have been tested with different DS1 datasets.

Relevant statistics are below.

The training and test datasets are different from each other. In the estimations made with Logistic Regression (LR) and Naive Bayes (NB) algorithms, very low results have been obtained compared to the others. The best results have been obtained with the Decision Tree (DT) algorithm. These Machine Learning (ML) classifiers do very well regarding False Positives. But the Deep Neural Network (DNN) gets the best results in the third experiment [33].

In another article, Botnet detection experiments have been carried out with Decision trees, Random Forest (RF), and k-NN models.

The authors have stated that the CTU-13 data set would give unbalanced results between classes.

Models based on Decision Trees have given the best results.

It has been also tested that adding artificial records to the datasets wouldn’t seriously affect the result.

It is stated in the article that it is not logical to use the IP address for the following reasons.

- Because IP addresses are automatically assigned, malware cannot affect this information.

- Because these IP addresses can be hidden by proxy or NAT service

- Because it will cause the IP address that the model has learned on any network to appear as a malicious machine on a different network.

As a result of the article, it is stated that K-NN has high resource usage and is 100 times slower than Decision Tree [34].

Hellinger Distance (HD) function is used in the traffic analysis phase. The J48 (Decision Tree) classifier, which has a classification accuracy of 99.67%, classifies DDoS packets. [35].

In the study, Random Forests is the algorithm that gives the best result among the Naive Bayes and Knn algorithms, with 99.68%. The test is done with a three-way handshake attack on TCP [36].

This research states that Machine Learning is used for botnet detection, but it is rarely done with artificial neural networks. In the conclusion part of the article, it is stated that botnet detection can be done with the artificial neural network method. The article comparing deep learning and machine learning states that deep learning reaches over 99.6% accuracy [37].

A botnet detection system using a multi-layered artificial intelligence framework with machine learning algorithms is presented. In the system, 2 modules detect botnet traffic, as shown in Figure 11.

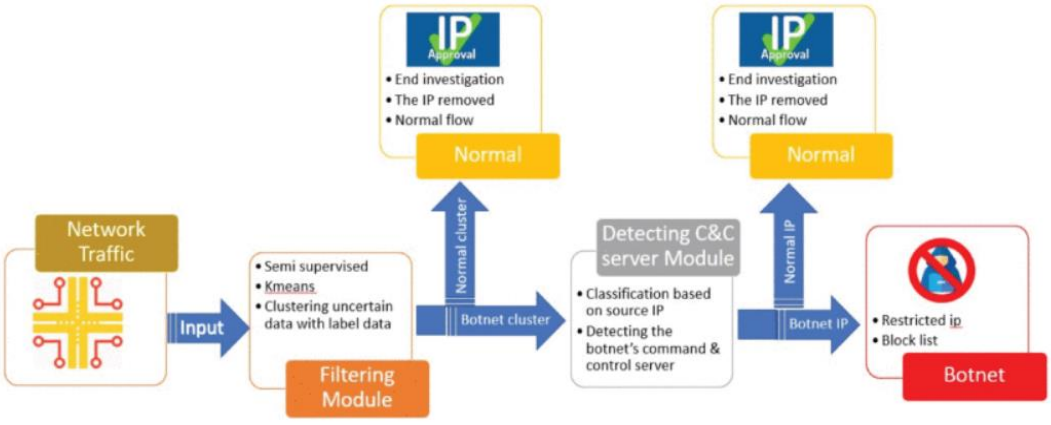

Figure 11: The Proposed Multilayer Structure for Botnet Detection Using Machine Learning Methods [1]

The behavior-based algorithm is used in Module 1, and the number of features is minimized. After this stage, normal tagged traffic has been removed from the list, and indefinite traffic has been transferred to module 2.

In the second module, 3 different algorithms have been used, namely Multilayer Perceptron (MLP), K-Nearest Neighbor (k-NN), and Support Vector Machine (SVM). This module aims to detect the C&C server and prevent the entry of the source IP address to the network.

When the research results have been examined, the accuracy values have been determined as K-NN 92.2%, SVM 85.03%, and MLP 84.58%. As a result, the authors determine that the K-NN algorithm gives the best result [1].

2.3 CONTRIBUTION

In order to detect DDOS attacks, which continue to increase their impact and validity in the IT environment, it is aimed to detect attacks in real-time by analyzing the behavior patterns of network packets. As a result of researching the studies and methods used in the literature on DDOS behavior detection, the studies with artificial intelligence are more valid. The most important factor in choosing this method is the necessity of making real-time detection and the attack open to manipulation.

- The main contribution of this study arises from the generation of new data that contains different traffics in the local environment. To compare the results the most used CTU-13 dataset has been used. CTU-13 dataset is a reliable dataset accepted in similar studies.
- Since the most crucial factor in Machine Learning is feature selection, we have conducted our tests with different feature selections. The virtual dataset(newly generated dataset) has been used in order to be more effective in feature selection. The virtual dataset has been detected by generating malicious and regular traffic between locally installed devices. Thus, dataset diversity has been provided with new package types. In this way, different scenarios have been compared by using the features that would be useful in the study. As a result of the comparisons, it has been discussed how useful the selected features are in pest detection.
- Generally accepted algorithms have been used in the studies, but parameter tuning has been done considering that the default settings could be improved further. Parameter tuning results are also shown in the thesis.

Another important factor is thought to be the feature selection in the datasets. While only the given features have been used in the CTU-13 dataset, the required features have been used in the virtual dataset.

CHAPTER III

ENVIRONMENTS

3.1 ENVIRONMENTS

The study aims to train certain models and increase the prediction results that will emerge with different datasets. In this study, two different datasets have been used; CTU-13, which is the most studied dataset in the literature, and the Virtual Datasets prepared to contain different traffics in the local environment.

3.1.1 CTU-13 Dataset

When similar articles have been examined, the 12th dataset is used most in the literature tested. This thesis conducts studies on data sets 2, 6, 9, 10, and 12. There are two types of file extensions inside each dataset group; pcap and binetflow extensions. Each data is labeled in the Binetflow file, and botnet traffic is clearly indicated. Since the number of features in CTU-13 datasets is limited, 7 features have been used.

3.1.2 Virtual Datasets

In addition to the CTU-13 data set, packet capture is performed with the environment of local devices. The first step is to install the devices and make a network connection between them. Five different machines have been created, and operating systems have been installed in the virtualization environment. A Linux operating system is installed on the victim machine to capture packets with "argus". All packet captures are made on the victim machine.

Then the goal is to create legal traffic in the real environment. For this purpose, IIS, Telnet, FTP, DHCP, etc., services have been installed on the devices, and communication between the victim machine and other devices is ensured. DHCP, ARP, etc., system packets on the victim machine have also been captured. Internet access has also been opened to capture different packet types.

Two different operating systems have been installed for attack purposes. The attack is made with Kali Linux and Windows machines used in security analysis. The attack is created with the low bandwidth dos tool "slowloris" on Kali Linux. The feature of the Slowloris tool is to avoid being caught by security devices, using as little bandwidth as possible. The attacker sends an incomplete HTTP request to the victim system and reserves system resources. After a while, the systems become unresponsive. On Windows, a TCP flood attack has been made. TCP packets of the desired size and quantity have been delivered to the victim's PC.

Apart from these, a Windows and a Linux machine have been created in order to create legal traffic. The purpose of these machines is to create traffic between them and the victim.

In order to make the captured packets more realistic, the ip segments of all devices have been changed and simulated. Thus, different types of traffic from many different IPs may be captured.

With the simulated environment, 21,247 instances have been captured. In network traffic capture software, each packet passing through the ethernet card is considered one instance. However, Argus groups similar packages and shows them in a single record. For this reason, the number of packages is more than the specified number.

3.1.3 Features Selection

Argus contains more than 120 features. These features can be used for different purposes. By trying many features, our test dataset has been obtained. When all features are examined, it has been concluded that 17 features contain meaningful data that can be used. In the study, different variations on 17 features have been examined.

3.1.4 Virtual Environment

In the virtual dataset, 6 scenarios have been created by adding and removing features. The scenarios and their contents are shown in Table 1 and described below. The aim here is to measure the effect of features on the prediction.

Table 1: Features of Scenario

	Scenerio 1	Scenerio 2	Scenerio 3	Scenerio 4	Scenerio 5	Scenerio 6
Label	X	X	X	X	X	X
Dur						X
Saddr	X	X	X			X
Daddr	X	X	X			X
Proto	X	X	X	X	X	X
Sport	X	X	X	X	X	X
Dport	X	X	X	X	X	X
Pkts						X
Bytes	X	X	X		X	X
Cause				X		X
Dttl						X
Dhops						X
Load			X	X		X
Dir			X	X		X
State			X	X		X
Smeansz		X			X	X
Dmeansz		X			X	X

Scenario 1:

The same features have been selected as the CTU-13 dataset used in the thesis. The CTU-13 dataset and the local dataset obtained are used for modeling and its effect on the models is examined.

Scenario 2:

The size of the packets is a feature that can help to find malicious traffic. The package contains the length of the data along with the default headers. In this scenario, the package size is added and the result is examined.

Scenario 3:

In this scenario, 3 additional features (Load, Dir, State) have been added that show the status of the packages. Since this flag is expected to have similar features for malicious traffic, it is thought to be helpful in estimation.

Scenario 4:

Although the IP address seems to be the most prominent feature, it is the value that is most open to manipulation. It has been examined what kind of result will be obtained without the IP address and the number of bytes.

Scenario 5:

In order to assess how the IP address affects scenario 2's outcome, the source IP address and destination IP address have been extracted for the features in scenario 2.

Scenario 6:

All available features have been used in this scenario. The important thing to remember is that algorithms will require more resources at a higher rate as the number of features grows.

3.1.5 Normalization

Data Cleaning

Missing Value: Data may be missing in some columns in the data set. When the datasets have been examined, it is determined that some values are recorded as blank.

Noisy Data: Data that is noisy is useless data. The phrase has frequently been used interchangeably with faulty data. When the datasets are inspected, it becomes clear that some values are entered for fields that should be integers but instead contain strings or otherwise useless data. This noisy data has been detected and fixed.

Inconsistent Data: The data contained in some attributes is beyond what is expected. Different formats in the feature have been converted to the appropriate format. For example, while the port number is an integer value, it appears in a hexadecimal format in some data. These values have been converted to binary format.

Feature Engineering

Normalize / Standardize: The major and minor values in the set have been standardized.

Sampling Data

Random Sampling: 30% of the data are used for testing, and 70% for training. The data have been randomly selected by the ML Tool.

Data Transformation

Attribute Selection: There are more than 120 attributes in the collected data and it has been reduced to 17 attributes. Different variations among 17 attributes are used in the scenarios. The attributes that are not thought to affect the results in the data sets have been removed. For example: "StartTime" indicates the start of the event. It is thought that it will not have any effect on the result.

One-Hot Encoding: One-hot encoding in machine learning is converting categorical information into a format that may be fed into machine learning algorithms to improve prediction accuracy. Since the direction property is categoric, it has been converted to a numeric value by one hot encoding.

Imbalanced Data

Downsampling Majority Class (Random Under Sampling etc.): The size of the CTU-13 datasets is quite large. Rows are deleted randomly in order to bring them to the desired number.

Balanced Class Weight: One of the popular techniques for unbalanced classification models is the balanced weight. In order to improve model performance, it adjusts the class weights of the majority and minority classes during the model training phase.

K-fold Cross Validation: When the dataset is folded K times, K-fold cross-validation takes place to evaluate how well the model performs when presented with fresh data. The data sample is separated into K groups, which equals the number of groups.

3.2 METHODS

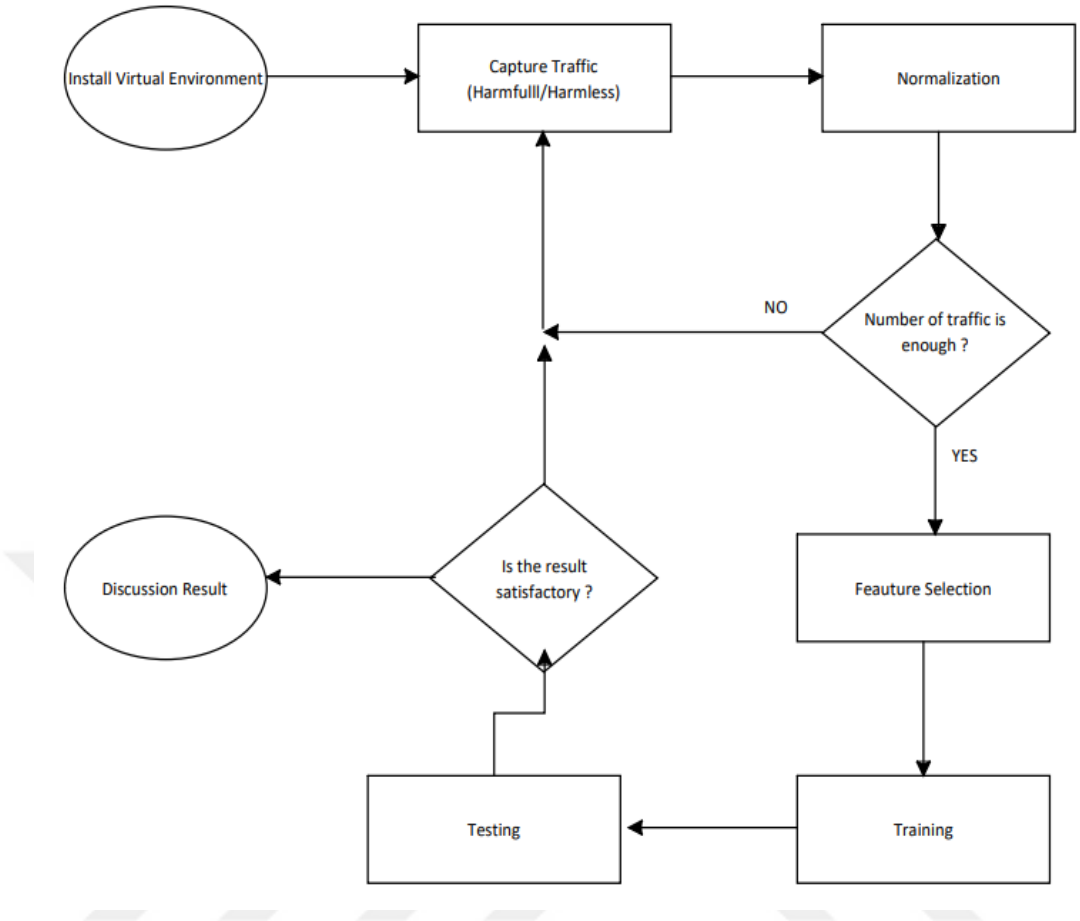

Figure 12: Data Set Preparation and Evaluation for Each Scenario

The algorithm starts with loading the local simulation environment and making the necessary configurations, as shown in Figure 12.

- After the environment is ready, the dataset is prepared. In order to prepare a data set, harmful and regular traffic must be generated in the environment and captured with appropriate tools.

- In the next step, necessary normalization operations should be done on the dataset. If a ready dataset is used, the previous steps are not applied.

After the normalization process, it is checked whether the dataset size is sufficient. If the amount of captured traffic is low, repeat the "Capture Traffic" step.

If the dataset size is sufficient, feature selection is made. Features are checked and unneeded features are removed.

The following 2 steps are the machine learning step. First, training and then testing phases are applied.

The test result is interpreted. If the result is not satisfactory, repeat the "Traffic Capture step" and continue by catching the new packet.

CHAPTER IV

EXPERIMENTAL RESULTS

4.1 EXPERIMENTAL RESULTS

This section presents the detection of malicious traffic on the simulated network on local devices and the CTU-13 dataset, and comparisons with other studies are given.

More than 60 models with a total of 7 different data sets and 5 different algorithms have been reported. The machine learning tool used generates 5 different scores for each model. As an evaluation metric, when different articles are examined, since CA (Classification Accuracy) is a generally accepted value, it has been used to compare the results in the thesis.

4.1.1 Hyperparameter Tuning

The purpose of machine learning is to train models with train datasets and to obtain high-accuracy results with the test dataset. Each model needs different parameters for operation. Machine Learning tools initially model with default parameter settings.

The default parameter values and the best results for each model are shown in Table 2, Table 3, Table 4, and Table 5.

Parameter Tunning Settings :

Table 2: Random Forest Parameters

Parameter Name	**Default Value**	**Value after Hyper Parameter Tunning**
Number of trees	10	10
Number of attributes considered at each split	No	5
Replicable training	No	No
Balance class distribution	No	No
Limit depth of individual trees	No	12
Do not split subsets smaller than	5	8

Table 3: KNN Parameters

Parameter Name	Default Value	Value after Hyper Parameter Tunning
Number of nearest neighbors	5	6
Metric	Euclidean	Manhattan
Weights	Uniform	Distance

Table 4: Decision Tree Parameters

Parameter Name	Default Value	Value after Hyper Parameter Tunning
Induce binary tree	Yes	Yes
Min. number of instances in leaves	2	2
Do not split subsets smaller than	5	5
Limit the maximal tree depth	100	100
Stop when majority reaches [%]:	95	100

Table 5: Support Vector Machine Parameters

Parameter Name	Default Value	Value after Hyper Parameter Tunning
SVM Cost	1	25,4
SVN ε	0,1	0,1
ν-SVM Cost	No	No
ν-SVM ν	No	No
Kernel	RBF g = auto	RBF g = 5.00
Numerical Tolerance	0,001	0,25
Iteration Limit	100	100

Table 6: The Results of Parameter Tunning in CTU-13 Dataset

Parameter Name	Accuracy (Default)	Accuracy (Parameter Tuning)	Increase (%)
Random Forest	0.9492	0.9938	4.70
K-NN	0.9805	0.9858	0.54
Decision Tree	0.9824	0.9940	1.18
Support Vector Machine	0.3280	0.8230	150.91

First of all, learning and test stages are operated with standard settings in each model. Then, the same stages have been applied by changing the values that the parameters can take for each model.

As seen in Table 6, the accuracy value has been increased a lot by parameter tuning for the Support Vector Machine.

However, an increase in accuracy has been observed in all cases, which showed the importance of parameter tuning.

4.1.2 CTU-13 Dataset Results

In Table 7, 5 datasets have been selected from the CTU-13 dataset with 7 features. Data normalization processes are performed for each data set. Random Under Sampling is performed so that there are 100,000 harmless traffic and 2,000 malicious traffic in each dataset. The best values are colored in Table 7.

Table 7: Results For CTU-13 Datasets.

Dataset	Model	AUC	CA	F1	Precision	Recall
CTU13-2	Random Forest	0.9989	0.9978	0.9977	0.9977	0.9978
CTU13-2	K-NN	0.9862	0.9967	0.9967	0.9967	0.9967
CTU13-2	Decision Tree	0.9700	0.9971	0.9971	0.9971	0.9971
CTU13-2	Naive Bayes	0.9889	0.9566	0.9673	0.9858	0.9566
CTU13-2	SVM	0.9870	0.9780	0.9710	0.9680	0.9780
CTU13-6	Random Forest	0.9986	0.9998	0.9998	0.9998	0.9998
CTU13-6	K-NN	0.9985	0.9993	0.9993	0.9993	0.9993
CTU13-6	Decision Tree	0.9964	0.9998	0.9998	0.9998	0.9998
CTU13-6	Naive Bayes	0.9955	0.9940	0.9942	0.9948	0.9940
CTU13-6	SVM	0.9990	0.9980	0.9991	1.0000	1.0000
CTU13-9	Random Forest	0.9983	0.9975	0.9975	0.9975	0.9975
CTU13-9	K-NN	0.9836	0.9962	0.9962	0.9962	0.9962
CTU13-9	Decision Tree	0.9636	0.9968	0.9968	0.9968	0.9968
CTU13-9	Naive Bayes	0.9584	0,9752	0.9745	0.9739	0.9752
CTU13-9	SVM	0.0710	0,9870	0.9860	0.9850	0.9870
CTU13-10	Random Forest	0.9996	0.9998	0.9998	0.9998	0.9998
CTU13-10	K-NN	0.9962	0.9985	0.9986	0.9986	0.9985
CTU13-10	Decision Tree	0.9992	0.9997	0.9997	0.9998	0.9997
CTU13-10	Naive Bayes	0.9845	0.9975	0.9975	0.9975	0.9975
CTU13-10	SVM	0.9440	0.9820	0.9790	0.9780	0.9820
CTU13-12	Random Forest	0.9996	0.9969	0.9969	0.9969	0.9969
CTU13-12	K-NN	0.9648	0.9893	0.9888	0.9885	0.9893
CTU13-12	Decision Tree	0.9661	0.9966	0.9965	0.9965	0.9966
CTU13-12	Naive Bayes	0.8429	0.9672	0.9663	0.9655	0.9672
CTU13-12	SVM	0.1290	0.9685	0.9770	0.9770	0.9820

The Random Forest algorithm is the most successful when the findings in Table 7 are analyzed, and the Naive Bayes approach is the least successful. The margin of error in the Random Forest model is between 3 per thousand and 0.2 per thousand. The factor that positively affects the Random Forest algorithm result is that there is sufficient data for the subtrees and most of the results obtained from the trees are correct. It can be concluded that this is the biggest factor in Random Forest's success.

K-NN and Decision Tree algorithms have also shown successful results. These algorithms also show that they can be used for the prediction of botnet behavior.

Naive Bayes gives very bad results without parameter tuning. Although there has been a serious improvement after parameter tuning, it takes the last place among the models. The feature of the Naive Bayes algorithm is that it creates a pattern among the data and compares this pattern with the test data. It can be concluded that the estimation result is not the best because it could not correctly establish the pattern between the pattern and the test data.

Table 8 shows the results of the data with a total of 7 features in terms of accuracy. Decision Tree gives the most accurate results among 5 malicious and 1 harmless traffic. It is concluded that the success of the Decision Tree is due to its ability to make more accurate predictions with the tree structure in multi-output problems.

In this scenario, the slightly less successful algorithm is the Naive Bayes algorithm.

Table 8: Multiple Regression Result For CTU-13 Datasets

Model	**AUC**	**CA**	**F1**	**Precision**	**Recall**
Random Forest	0.9988	0.9922	0.9921	0.9920	0.9922
K-NN	0.9844	0.9820	0.9820	0.9819	0.9820
Decision Tree	0.9851	0.9924	0.9924	0.9924	0.9924
Naive Bayes	0.9643	0.8968	0.9079	0.9303	0.8968
SVM	0.5670	0.9020	0.8720	0.8470	0.9020

For multiple regression, 5 different malicious traffic groups have been selected. It has been simplified to 20,000 harmless traffic and 2,000 malicious traffic from each group. Then, 5 data sets have been combined and a data set with 5 malicious and 1 harmless traffic type emerged. The expectation from the models is to make predictions about which group the traffic type falls into. Table 8 shows the estimation results made in the models.

When the results have been examined, successful results have been obtained with Random Forest and Decision Tree algorithms. The slightly less successful results have been obtained with Naive Bayes. It can be noted that the results are similar to the results in Table 7.

Table 9: Random Forest Confusion Matrix

	Predicted							
Actual	Neris	Rbot	Virut	Menti	Murlo	Nsis	Harmless	Total
Neris	895	30	85	25	0	11	24	1070
Rbot	16	1013	0	1	0	6	8	1044
Virut	24	3	966	17	0	21	27	1058
Menti	57	1	50	948	1	2	0	1059
Murlo	6	6	0	3	988	18	4	1025
Nsis	7	7	3	0	1	1052	15	1085
Harmless	13	4	30	0	0	1	6211	6259
Total	1018	1064	1134	994	990	1111	6289	12600

Table 10: KNN Confusion Matrix

	Predicted							
Actual	Neris	Rbot	Virut	Menti	Murlo	Nsis	Harmless	Total
Neris	913	28	66	40	1	8	14	1070
Rbot	17	1009	2	1	0	5	10	1044
Virut	60	4	929	38	2	15	10	1058
Menti	57	1	40	958	0	2	1	1059
Murlo	5	6	2	3	997	11	1	1025
Nsis	11	6	19	0	14	991	44	1085
Harmless	31	58	29	5	9	40	6087	6259
Total	1094	1112	1087	1045	1023	1072	6167	12600

Table 11: Decision Tree Confusion Matrix

	Predicted							
Actual	Neris	Rbot	Virut	Menti	Murlo	Nsis	Harmless	Total
Neris	925	30	50	33	6	6	20	1070
Rbot	16	1013	1	2	1	4	7	1044
Virut	86	3	904	38	1	13	13	1058
Menti	53	1	41	962	0	1	1	1059
Murlo	5	7	4	1	995	9	4	1025
Nsis	10	12	11	4	9	1028	11	1085
Harmless	33	4	29	2	3	9	6179	6259
Total	1128	1070	1040	1042	1015	1070	6235	12600

Table 12: Naive Bayes Confusion Matrix

	Predicted							
Actual	Neris	Rbot	Virut	Menti	Murlo	Nsis	Harmless	Total
Neris	662	0	60	168	35	10	135	1070
Rbot	16	957	1	0	0	12	58	1044
Virut	371	3	138	205	17	2	322	1058
Menti	71	0	9	947	26	0	6	1059
Murlo	144	0	35	1	659	9	177	1025
Nsis	145	5	26	0	4	633	272	1085
Harmless	190	86	68	10	272	474	5159	6259
Total	1599	1051	337	1331	1013	1140	6129	12600

Table 13: SVM Confusion Matrix

	Predicted							
Actual	Neris	Rbot	Virut	Menti	Murlo	Nsis	Harmless	Total
Neris	221	6	40	497	45	140	121	1070
Rbot	285	700	0	0	0	59	0	1044
Virut	534	0	170	97	2	158	97	1058
Menti	197	1	139	719	0	3	0	1059
Murlo	402	0	0	45	541	27	10	1025
Nsis	853	2	0	0	0	159	71	1085
Harmless	5969	4	0	3	2	23	258	6259
Total	8461	713	349	1361	590	569	557	12600

Table 14: Multiple Classification Result For CTU-13 Datasets

	Random Forest	KNN	Decision Tree	Naive Bayes	SVM
Neris	0,8364	0,8533	0,8645	0,6187	0,2065
Rbot	0,9703	0,9665	0,9703	0,9167	0,6705
Virut	0,9130	0,8781	0,8544	0,1304	0,1607
Menti	0,8952	0,9046	0,9084	0,8942	0,6789
Murlo	0,9639	0,9727	0,9707	0,6429	0,5278
Nsis	0,9696	0,9134	0,9475	0,5834	0,1465
Harmless	0,9923	0,9725	0,9872	0,8243	0,0412
Average	0,9582	0,9432	0,9529	0,7266	0,2197

Multiple regression results have been also examined in the study. The aim of multiple regression is to distinguish the types of malware in malicious traffic. A dataset consisting of 6 malicious (9.000 instances) and 1 harmless (9.000 instances) traffics in the CTU-13 dataset has been created. Confusion Matrix results for Random Forest, KNN, Decision Tree, Naive Bayes and SVM algorithm are shown respectively in Table 9, Table 10, Table 11, Table 12 and Table 13. Table 14 gives the overall results.

When the results are examined, it is seen that the results of Random Forest and Decision Tree algorithms are successful, similar to the other tables. Again, similar to the other tables, regression results are unsatisfactory for Naive Bayes and Support Vector Machine algorithms.

While the algorithms have been generally successful in classifying the Rbot and Menti malware, they fail in classifying the Neris and Virut malware.

Table 15: Stacking Table

	Naive Bayes	**SVM**	**Stack Multiple Model**
CTU13-2	0,9566	0,9780	0,9802
CTU13-6	0,9940	0,9980	0,9995
CTU13-9	0,9752	0,9870	0,9828
CTU13-10	0,9975	0,9820	0,9976
CTU13-12	0,9672	0,9686	0,9805

The ensemble method is used to produce a single optimum predictive model as a result of multiple models working together.

It is used as a "Stacking Model" in the ML tool used in the study. The two algorithms (Naive Bayes and Support Vector Machine) that gives the worst test results have been run together in the Stacking Model. The results are shown in Table 15.

Examining the findings reveals that the stacking results in four groups outperform those of the 2 algorithms. The observed results show that the Ensemble method has a positive effect on the results. Nevertheless, the stacking model results haven't changed its place in the ranking.

4.1.3 Virtual Environments Dataset Results

In the virtual dataset, 6 different scenarios have been created by adding and removing features. The scenarios and their contents are specifically described in Table 1.

The values shown in Table 16 are the average results of 5 models obtained using different features in the dataset created in the virtual environment. There are 102,000 instances in each scenario.

Table 16: Regressions For Virtual Datasets Scenarios

SCENARIOS	AUC (average)
Scenario – 1	0.96548
Scenario – 2	0.96370
Scenario – 3	0.98988
Scenario – 4	0.99142
Scenario – 5	0.96371
Scenario – 6	0.99821

The purpose of modeling with different features is to see how the features affect the result. The features of the scenarios are stated above.

In scenario 6, 17 features that can be used for botnet prediction are used. The best prediction result is achieved in this scenario. The use of all features has increased the success of the models in decision-making. However, it has also negatively affected the use of resources. Since all the features in the data set are processed in the models, it increases the processor load and causes the result to be formed later.

Scenario 1 uses the same features found in the CTU-13 dataset. For this reason, scenario 1 has been used as a reference. The estimation has been made with the features used for CTU-13 has taken its place at the end.

Since the IP address is an important feature in the network, the features related to the IP address are also examined in the scenarios. A good result has been obtained in the 4th scenario, where the IP address is not used and the features related to the status of the packets are used. Scenario 2 and scenario 5 are also set as good examples to examine the IP address. Their difference has been whether the IP address is used or not. Similar results have been obtained in both scenarios.

Although the IP address is the most important property for network packets, it has been seen that the IP address is not an important factor in botnet detection using machine learning algorithms.

For scenario 3, properties showing the status of the network packet are taken into account. Although better results have been obtained than the reference scenario, it is not among the best.

It is observed that the results for scenario 2 are also in the lower ranks. Although packet size characteristics are also taken into account in Scenario 2, packet size appears to lead to misleading results for malware detection.

4.1.4 Compare to Previous Studies

Similar scenario studies have been examined in the literature, and the outcomes are shown in Table 17.

Table 17: Comparison of median CA scores in the CTU-13 dataset with other studies

Other Studies	Random Forest	K-NN	Decision Tree	Naive Bayes	SVM
Botnet Attack Detection using Machine Learning [38]		91	99.91	97.10	100
Analyzing Machine Learning-based Feature Selection for Botnet Detection [39]	96.9	99.56	99.18	99.5	89.68
Botnet Detection Approach Using Graph-BasedMachine Learning [40]		99	99	95	
Deep learning-based classification model for botnet attack detection [37]			95.2	98.5	99.5
An adaptive multi-layer botnet detection technique using machine learning classifiers [41]		93.9		75	
Multilayer Framework for Botnet Detection Using Machine Learning Algorithms [1]		92.2			85.95
This Study	99.84	99.60	99.80	97.81	98.54

The common points of the studies are that they have been studied with the CTU-13 dataset and the CA (Classification Accuracy) value is used for the evaluation results.

In our study, 5 different dataset groups have been taken from CTU-13, and the averages of CA scores have been reflected in Table 17.

Compared to Table 17 results;

First of all, as a summary of the study, it may be said that the botnet detection algorithms Random Forest, K-NN, and Decision Tree work well.

Similar results are seen among studies, except for Ali Ahmed et al. [42].

The Random Forest algorithm, which emerged as the most successful algorithm, has not been studied by most researchers. The reason is estimated to be the advanced version of the Decision Tree algorithm. However, according to our experiments, it gives better results than the Decision Tree algorithm.

There are differences between studies when the least unsuccessful algorithms are considered.

Considering the total scores (100% accuracy results are not evaluated.) Alshamkhany et al. show the result of 99.91. This result has been obtained with the Decision Tree algorithm. In the next order, there is our work with 3 algorithms. This shows that our study is not based on modeling, but the normalization of the dataset, the selection of parameters related to the models and the test performed are more accurate than the others.

CHAPTER V

CONCLUSION

5.1 CONCLUSION

This study, it is aimed to detect DDOS attacks, which are one of the biggest dangers of IT infrastructure. This approach aims to examine packets at the network level and to classify them correctly with artificial intelligence. The main motivation of the study is algorithms, feature selection, and parameter selection.

In the study, the CTU-13 dataset and the dataset created in the local environment have been used. With the globally accepted CTU-13 data set, the level of the study has been increased and it has been possible to compare it with other studies. Studies have been completed on 5 different groups within the data set.

In the locally created dataset, 17 features that can be used for the study have been selected and the effects of different variations and features on the result have been discussed.

Parameter selection is an important issue for ML algorithms. The ideal parameter set must be chosen in order to improve the result's accuracy. For this purpose, parameter tuning has been made at the beginning of the study and continued with the determined parameter set.

In the second part, a study has been conducted on 5 different malicious traffic groups selected in the CTU-13 dataset. Data cleaning (missing or noisy data, outlier value regulation, etc.), data transformation (attribute selection, min-max normalization, etc.), and data balancing (undersampling data, the balanced weight, etc) processes are applied to each dataset.

Then, the data sets have been processed with 5 algorithms and the results have been examined. It has ranked at the top of the Random Forest algorithm with a rate of 99.84%. In second place is the Decision Tree algorithm with a rate of 99.80%. Similarly, multiple regression has been applied to a data set with all malicious data. The aim of this study is to find harmful or harmless traffic in the dataset formed as a

result of the combination of 5 different malware. As a result of multiple regression, Decision Tree, and Random Forest algorithms gives the best results. These two algorithms are a continuation of each other. The source of success is the Decision Tree algorithm. A Decision Tree splits large amounts of data and establishes simple decision mechanisms. They have also been successful in detecting malicious traffic. The Random Forest is made up of a variety of decision trees. The algorithm with the worst results in the study is the Naive Bayes algorithm. Naive Bayes calculates a probability for each element. When calculating probability, each feature is considered independent of each other. It is thought that it gives the worst result because the relations between the variables cannot be modeled.

In addition to these, studies on multiple classification and stacking have also been carried out. In the multiple classification study, it is aimed to accurately predict the type of pest in 6 different malicious and harmless traffic. In parallel with other studies in multiple classification, Random Forest and Decision Tree algorithm results were seen as successful. In addition, Naive Bayes and Support Vector Machine algorithm results are seen as unsuccessful. In the stacking study, the 2 most unsuccessful algorithms have been used on a common model. It has been observed that the algorithms give better results when used together. However, it has been found to be unsuccessful compared to other algorithms.

In the third chapter, using the features taken from the local dataset, experiments have been made with different features in 6 different scenarios. It has been tried to examine how the properties used affect the result.

It has been determined that the features given in the CTU-13 dataset are limited and the accuracy rate of the estimation made with these limited features is low.

Working with 17 features determined as available gives the best results, but taking into account the volume of the dataset and the amount of computing power needed for the calculations, it is estimated that it will require a serious resource.

Another important result is that the scenario where the IP address, which is the most important feature of the network packets, is not used gives better results. It can be concluded that using the IP address does not make a positive contribution.

5.2 FUTURE WORK

The article includes studies on understanding DDOS behaviors. It is desired to detect the attack during or at the beginning of a DDOS attack and to establish mechanisms that prevent it afterward. Although it is thought that the most appropriate parameters are selected in the algorithms, it is thought that changes in the parameters may have a positive effect on the result.

However, the study proceeded through an existing dataset. In real life, a dataset will not be available for a DDOS attack. First of all, a DDOS attack will be detected, then the anomaly traffic will be tagged and the data set will begin to form. Because of this, service interruptions are possible.

In the study, the Argus network analysis tool has been used for traffic analysis. The study has been carried out on the features provided by the Argus software. As a result of testing with different network analysis tools, positive changes can be observed in the results by using different features.

Feature selection is also among the factors that positively affect the result. More detailed feature selection and results can be examined.

REFERENCES

[1] IBRAHIM Wan Nur Hidayah, ANUAR Syahid, SELAMAT Ali, KREJCAR Ondrej, CRESPO Rubén González, HERRERA VIEDMA Enrique and FUJITA Hamido (2021), "Multilayer Framework for Botnet Detection Using Machine Learning Algorithms", *IEEE Access*, Vol. 9, pp. 48753-48768.

[2] GAURAV Akshat (2022), "A Comprehensive Survey on DDoS Attacks on various Intelligent Systems and It's Defense Techniques", *International Journal of Intelligent Systems,* Vol. 37, pp. 11407-11431.

[3] LOHACHAB Ankur and KARAMBIR Bidhan (2018), "Critical Analysis of DDoS-An Emerging Security Threat over IoT Networks", *Journal of Communications and Information Networks*, Vol. 3, pp. 57-78.

[4] ALOMARI Esraa, MANICKAM Selvakumar, KARUPPAYAH Shankar and GUPTA Brij (2012), "Botnet-Based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art", *International Journal of Computer Applications,* Vol. 49, pp. 24-32.

[5] GUVEN Erhan and BUCZAK Anna (2015), "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection", *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 2, pp. 1153-1176.

[6] DIMOLIANIS Marinos, PAVLIDIS Adam and MAGLARIS Vasilis (2021), "Signature-Based Traffic Classification and Mitigation for DDoS Attacks Using Programmable Network Data Planes", *IEEE Access,* Vol. 9, pp. 113061-113076.

[7] MASDARI Mohammad and KHEZRI Hemn (2020), "A survey and taxonomy of the fuzzy signature-based Intrusion Detection Systems", *Applied Soft Computing*, Vol. 92, pp. 106301.

[8] SALEM Fatty, YOUSSEF Hoda, ALI Ihab and HAGGAG Ayman (2022), "A variable-trust threshold-based approach for DDOS attack mitigation in software defined networks", *Plos One,* Vol. 17, No. 8, pp. e0273681.

[9] VERMA Priyanka, TAPASWI Shashikala and GODFREY Wilfred (2020), "An Adaptive Threshold-Based Attribute Selection to Classify Requests Under DDoS Attack in Cloud-Based Systems", *Arabian Journal for Science and Engineering*, Vol. 45, pp. 2813-2834.

[10] WEI Wei, CHEN Feng, XIA Yingjie and JIN Guang (2013), "A Rank Correlation Based Detection against Distributed Reflection DoS Attacks", *IEEE Communications Letters*, Vol. 17, No. 1, pp. 173-175.

[11] BHUYAN Monowar, GOSWAMI A., BHATTACHARYYA D.K. and KALITA J.K. (2015), "Low-Rate and High-Rate Distributed DoS Attack Detection Using Partial Rank Correlation", *2015 Fifth International Conference on Communication Systems and Network Technologies*, pp. 706-710, Gwalior, India.

[12] THAPNGAM Theerasak, YU Shui, ZHOU Wanlei and MAKKI S. Kami (2014), "Distributed Denial of Service (DDoS) Detection by Traffic Pattern", *Peer-to-peer networking and applications*, Vol. 7, pp. 346-358.

[13] ZHAO David, TRAORE Issa, GHORBANI Ali, SAYED Bassam, SAAD Sherif and LU Wei (2012), "Peer-to-peer Botnet Detection Based on Flow Intervals", *Information Security and Privacy Research: 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012*, pp. 87-102, Berlin, Heidelberg.

[14] WITTEN Ian, FRANK Eibe and HALL Mark (2016), "Data Mining: Practical Machine Learning Tools and Techniques", *Acm Sigmod Record*, Vol. 31, No. 1, pp. 76-77.

[15] PRIYA S., SIVARAM M. and YUVARAJ D. (2020), "Machine Learning based DDOS Detection", *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*, pp. 234-237, Pune, India.

[16] HE Zecheng, ZHANG Tianwei and LEE Ruby (2017), "Machine Learning Based DDoS Attack Detection from Source Side in Cloud", *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 114-120, New York, USA.

[17] SHARAFALDIN Iman, LASHKARI Arash Habibi, HAKAK Saqib and GHORBANI Ali (2019), "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1-8, Chennai, India.

[18] NANDI Suman, PHADIKAR Santanu and MAJUMDER Koushik (2020), "Detection of DDoS Attack and Classification Using a Hybrid Approach", *2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP)*, pp. 41-47, Guwahati, India.

[19] DAYAL Neelam and SRIVASTAVA Shashank (2017), "Analyzing the behavior of DDoS attacks to identify DDoS detection features in SDN", *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*, pp. 274-281, Bengaluru, India.

[20] ABUROMMAN Abdulla Amin and REAZ Mamun Bin Ibne (2016), "A Survey of Intrusion Detection Systems Based on Ensemble and Hybrid Classifiers", *Computers & Security*, Vol. 65, pp. 135-152.

[21] YUAN Xiaoyong, LI Chuanhuang and LI Xiaolin (2017), "DeepDefense: Identifying DDoS Attack via Deep Learning", *2017 IEEE international conference on smart computing (SMARTCOMP)*, pp. 1-8, Hong Kong, China.

[22] DAS Saikat, MAHFOUZ Ahmed, VENUGOPAL Deepak and SHIVA Sajjan (2019), "DDoS Intrusion Detection Through Machine Learning Ensemble", *2019 IEEE 19th international conference on software Quality, Reliability and Security Companion (QRS-C)*, pp. 471-477, Sofia, Bulgaria.

[23] TUAN Tong Anh, LONG Hoang Viet, SON Le Hoang, KUMAR Raghvendra, PRIYADARSHINI Ishaani and SON Nguyen Thi Kim (2020), "Performance evaluation of Botnet DDoS attack detection using machine learning", *Evolutionary Intelligence*, Vol. 13, pp. 283-294.

[24] HUSSAIN Faisal, ABBAS Syed Ghazanfar, PIRES Ivan Miguel, TANVEER Sabeeha, FAYYAZ Ubaid, GARCIA Nuno and SHAH Ghalib (2021), "A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks", *IEEE Access*, Vol.9, pp. 163412-163430.

[25] WANG Haofan (2022), "Botnet Detection via Machine Learning Techniques", *2022 International Conference on Big Data, Information and Computer Network (BDICN)*, pp. 831-836, Sanya, China.

[26] KUMARI Kimmi and MRUNALINI M. (2022), "Detecting Denial of Service attacks using machine learning algorithms", *Journal of Big Data*, Vol. 9, No. 1, pp. 1-17.

[27] SINGH Ritik Raj, KAMILA Kaunik and KALAIVANI J. (2021), "Neural Network Based Botnet Detection", *2021 International Conference on Intelligent Technologies (CONIT)*, pp. 1-5, Hubli, India.

[28] YAMAGUCHI Shingo (2021), "A Basic Command and Control Strategy in Botnet Defense System", *2021 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1-5, Las Vegas, USA.

[29] HASAN Nasimul, CHEN Zhenxiang, ZHAO Chuan, ZHU Yuhui and LIU Cong (2022), "IoT Botnet Detection Framework from Network Behavior based on Extreme Learning Machine", *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1-6, New York, USA .

[30] KHALAF Bashar Ahmed, MOSTAFA Salama, MOHAMMED Mazin Abed, ABDUALLAH Wafaa Mustafa and MUSTAPHA Aida (2019), "Comprehensive Review of Artificial Intelligence and Statistical Approaches in Distributed Denial of Service Attack and Defense Methods", *IEEE Access*, Vol. 7, pp. 51691-51713.

[31] SOOD Isha and SHARMA Varsha (2021), "Computational Intelligent Techniques To Detect DDOS Attacks : A Survey", *Journal of Cybersecurity*, Vol. 3, No. 2, pp. 89.

[32] RAMOS Katherinne Shirley Huancayo, MONGE Marco Antonio Sotelo and VIDAL Jorge Maestre (2020), "Benchmark-Based Reference Model for Evaluating Botnet Detection Tools Driven by Traffic-Flow Analytics", *Sensors*, Vol. 20, No. 16, pp. 4501.

[33] SRINIVASAN Sriram, RAVI Vinayakumar, ALAZAB Mamoun and KP Soman (2020), "Network Flow-based IoT Botnet Attack Detection using Deep Learning", *IEEE INFOCOM 2020-IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pp. 189-194, Toronto, Canada.

[34] MATA Javier Velasco, CASTRO Víctor González, FIDALGO Eduardo and ALEGRE Enrique (2021), "Efficient Detection of Botnet Traffic by features selection and Decision Trees", *IEEE Access*, Vol. 9, pp. 120567-120579.

[35] BHAROT Nitesh, VERMA Priyanka, SHARMA Sangeeta and SURAPARAJU Veenadhari (2018), "Distributed Denial-of-Service Attack Detection and Mitigation Using Feature Selection and Intensive Care Request Processing Unit", *Arabian Journal for Science and Engineering*, Vol. 43, pp. 959-967.

[36] MISHRA Anupama, GUPTA B. B., PERAKOVIĆ Dragan, PEÑALVO Francisco José García and HSU Ching Hsien (2021), "Classification Based Machine Learning for Detection of DDoS attack in Cloud Computing", *2021 IEEE international conference on consumer electronics (ICCE)*, pp. 1-4, Las Vegas, USA .

[37] AHMED Abdulghani Ali, JABBAR Waheb A., SADIQ Ali Safaa and PATEL Hiran (2020), "Deep Learning-based Classification Model For Botnet Attack Detection", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 102, pp. 1-10.

[38] ALSHAMKHANY Mustafa, ALSHAMKHANY Wisam, MANSOUR Mohamed, KHAN Mueez, DHOU Salam and ALOUL Fadi (2020), "Botnet Attack Detection using Machine Learning", *2020 14th International Conference on Innovations in Information Technology (IIT)*, pp. 203-208, Al Ain, United Arab Emirates.

[39] SAFITRI Winda Ayu, AHMAD Tohari and HOSTIADI Dandy Pramana (2022), "Analyzing Machine Learning-based Feature Selection for Botnet Detection", *2022 1st International Conference on Information System & Information Technology (ICISIT)*, pp. 386-391, Yogyakarta, Indonesia.

[40] ALHARBI Afnan and ALSUBHI Khalid (2021), "Botnet Detection Approach Using Graph-Based Machine Learning", *IEEE Access*, Vol. 9, pp. 99166-99180.

[41] KHAN Riaz Ullah, ZHANG Xiaosong, KUMAR Rajesh, SHARIF Abubakar, GOLILARZ Noorbakhsh Amiri and ALAZAB Mamoun (2019), "An Adaptive Multi-Layer Botnet Detection Technique Using Machine Learning Classifiers", *Applied Sciences*, Vol. 9, No. 11, pp. 2375.

[42] AL QEREM Ahmad, ALAUTHMAN Mohammad, ALMOMANI Ammar and GUPTA B. B. (2020), "IoT transaction processing through cooperative concurrency control on fog–cloud computing environment", *Soft Computing*, Vol. 24, pp. 5695-5711.

[43] SHIROL Sujan, SAYEDI Husna, IRIONDO Roberto (2021), *K-Nearest Neighbors (KNN) Algorithm Tutorial - Machine Learning Basics*, https://pub.towardsai.net/k-nearest-neighbors-knn-algorithm-tutorial-machine-learning-basics-ml-ec6756d3e0ac, DoA. 01.03.2023.

[44] Larhmam (2018), *Support Vector Machine,* https://en.wikipedia.org/wiki/File:SVM_margin.png, DoA. 01.03.2023.

[45] MLMath.io (2019), *Math behind SVM (Support Vector Machine),* https://ankitnitjsr13.medium.com/math-behind-svm-support-vector-machine-864e58977fdb, DoA. 01.03.2023.

[46] AGARWAL Dishaa (2021), *Introduction to SVM(Support Vector Machine) Along with Python Code,* https://www.analyticsvidhya.com/blog/2021/04/insight-into-svm-support-vector-machine-along-with-code, DoA. 01.03.2023.