Abdurrahman KÜÇKKOÇ

ANALYSING NETWORK TRAFFIC AND DETECTING NETWORK THREATS BY USING THE ALGORITHMS OF MACHINE LEARNING

M.Sc.

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF ELECTRICAL AND COMPUTER ENGINEERING

By

Abdurrahman Küçükkoç

April 2024

M.Sc. Thesis

AGU 2024

ANALYSING NETWORK TRAFFIC AND DETECTING NETWORK THREATS BY USING THE ALGORITHMS OF MACHINE LEARNING

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
ELECTRICAL AND COMPUTER ENGINEERING

By
Abdurrahman Küçükkoç
April 2024

SCIENTIFIC ETHICS COMPLIANCE

I hereby declare that all information in this document has been obtained in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name-Surname: Abdurrahman Küçükkoç
Signature :

REGULATORY COMPLIANCE

M.Sc. thesis titled “Analysing network traffic and detecting network threats by using the algorithms of machine learning” has been prepared in accordance with the Thesis Writing Guidelines of the Abdullah Gül University, Graduate School of Engineering & Science.

Prepared By
Abdurrahman Küçükkoç

Advisor
Associate Professor Zafer Aydın

Head of the Electrical and Computer Engineering Program
Assist. Prof. Samet GÜLER

ACCEPTANCE AND APPROVAL

M.Sc. thesis titled “Analysing network traffic and detecting network threats by using the algorithms of machine learning” and prepared by Abdurrahman Kçükkoç has been accepted by the jury in the Electrical and Computer Engineering Graduate Program at Abdullah Gül University, Graduate School of Engineering & Science.

…….. /…….. / ……..

(Thesis Defense Exam Date)

JURY:

Advisor : Assoc. Prof. Zafer AYDIN…….....

Member : Assist. Prof. Abdulkadir KÖSE…….....

Member : Assist. Prof. Yasin GÖRMEZ…..…….....

APPROVAL:

The acceptance of this M.Sc. thesis has been approved by the decision of the Abdullah Gül University, Graduate School of Engineering & Science, Executive Board dated ….. /….. / …….. and numbered ……..……. .

…….. /…….. / ……..

(Date)

Graduate School Dean
Prof. İrfan ALAN

ABSTRACT

ANALYSING NETWORK TRAFFIC AND DETECTING NETWORK THREATS BY USING THE ALGORITHMS OF MACHINE LEARNING

Abdurrahman Küçükkoç
MSc. in Electrical and Computer Engineering
Advisor: Associate Professor Zafer Aydın

April 2024

As information technologies progress, the possibilities of access to information increase and therefore it becomes difficult to ensure the security of information. Today, with the use of information systems in all areas of life, network threats have also increased. The increase in individual access to and use of the internet has also brought network threats. In addition, the latest developments in information technologies, developing global communication networks, the internet of things aiming to connect all objects with networks, cloud technologies, the spread of mobile internet and the renewal of devices have brought network threats and uncertainties. Network threats increase the security vulnerabilities in the information and communication systems of individuals and organisations day by day. This situation causes systems to malfunction, economic damage and cyber security to be jeopardised. In order to contribute to individuals, institutions and organisations, our thesis aims to protect information systems against network threats, to ensure data confidentiality, integrity and accessibility, to detect network threats in advance and to take measures against these threats. We believe that by analysing heterogeneous network traffic, which includes most network attacks on the Internet, and using machine learning algorithms, we will reach a result close to reality in the detection of network threats. In line with this result, we will be able to take precautions against network threats in information systems and structures.

Keywords: Network threats, Machine Learning, Security vulnerabilities

ÖZET

MAKİNE ÖĞRENİMİ ALGORİTMALARINI KULLANARAK AĞ TRAFİĞİNİ ANALİZ ETME VE AĞ TEHDİTLERİNİ TESPİT ETME

Abdurrahman Küçükkoç
Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans
Tez Danışmanı: Doç. Dr. Zafer Aydın
Nisan - 2024

Bilgi teknolojileri ilerledikçe bilgiye erişim imkânları artmakta ve dolayısıyla da bilginin güvenliğinin sağlanması zorlaşmaktadır. Günümüzde bilişim sistemlerinin hayatın her alanında kullanılmaya başlanması ile birlikte ağ tehditleri de artmıştır. Bireysel olarak internete erişimin ve internet kullanımının artması ağ tehditlerini de beraberinde getirmiştir. Ayrıca bilişim teknolojilerindeki son gelişmeler, gelişen global iletişim ağları, tüm nesnelerin ağlarla birbirine bağlanmasını hedefleyen nesnelerin interneti, bulut teknolojileri, mobil internetin yaygınlaşması ve cihazların yenilenmesi ile birlikte ağ tehditleri ve belirsizlikleri de beraberinde getirmiştir. Ağ tehditleri bireyleri, kurumların bilgi ve iletişim sistemlerinde güvenlik zafiyetlerini her geçen gün arttırmaktadır. Bu durum sistemlerin çalışmamasına, ekonomik zarara ve siber güvenliğin tehlikeye girmesine neden olmaktadır. Birey, kurum ve kuruluşlara katkı sağlaması adına tez çalışmamız ağ tehditlerine karşı bilgi sistemlerini korumayı, veri gizliliğini, bütünlüğünü ve erişilebilirliğini güvence altına almayı, ağ tehditlerini önceden tespit etmeyi ve bu tehditlere karşı önlem almayı hedeflemektedir. İnternet üzerindeki çoğu ağ saldırılarını içeren heterojen ağ trafiğini analiz ederek ve makine öğrenmesi algoritmalarını kullanarak ağ tehditlerinin tespitinde gerçeğe yakın bir sonuca ulaşacağımıza inanıyoruz. Bu sonuç doğrultusunda bilişim sistem ve yapılarında ağ tehditlerine karşı önlemler almayı sağlayacağını düşünmekteyiz.

Anahtar kelimeler: Ağ saldırıları, Makine öğrenme, Güvenlik açıklıkları

Acknowledgements

I'd like to start by thanking my thesis advisor, Assoc. Prof. Dr. Zafer AYDIN, for taking my wishes into consideration and helping me choose the thesis topic. I'd also like to thank all the academic and administrative staff at Abdullah Gül University for contributing to my master's degree education. I'd like to thank my wonderful thesis defence jury members, Dr. Yasin GÖRMEZ and Dr. Abdulkadir KÖSE, for their invaluable help and guidance during the evaluation of my thesis study. I'd also like to thank my loving family for their unwavering support and encouragement throughout my entire education journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BHOs	Browser Helper Objects
CTR	Click-Through Rate
ISS	Internet Security Systems
DDoS	Distributed Denial-of-Service
XSS	Cross-Site Scripting
OS	Operating System
SSI	Server-Side Include
SVM	Support Vector Machine
XGBoost	eXtreme Gradient Boosting
GBM	Gradient Boosting Machine
IDEs	Integrated Development Environments
LightGBM	Light Gradient Boosting Machine
IDEs	Integrated Development Environments

To My Family

Chapter 1

Introduction

The purpose of this investigation is to analyse network traffic utilising machine learning algorithms for the detection of potential network threats. Network attacks may have various adverse effects including the theft of information, system shutdown, data alterations, service disruption, and intrusion upon user privacy. In modern-day society, a significant amount of confidential and sensitive data is stored on the internet and can be accessed through various networks. Therefore, businesses and organisations must prioritize robust network security measures to protect their data and limit access to only authorized personnel and partners. Developing network defence systems to mitigate and neutralise security risks encountered in information system network traffic will ensure the security of critical data types. This helps to prevent national security threats and public disorder. The implementation of these systems is a vital contribution to overall network security. We analysed the current network traffic and employed well-known algorithms including Light Gradient Boosting Machine, Support Vector Machine, K-Nearest Neighbours, XGBoost and Random Forests to determine the most appropriate model. The study comprises of four sections. The first section elucidates network attacks and their significance, while providing an overview of network attack types and relevant information. The second section elucidates machine learning algorithms and hyperparameter optimisation, with an emphasis on employing Python Programming Language as the application environment for the algorithms. The third section details the network dataset used and the models applied. Finally, the fourth section measures and analyses the diagnosis of machine learning algorithms used to analyse network traffic and detect network threats with the collected data, and evaluates the results.

Chapter 2

General Information

2.1 Network Attacks

A digital network attack may be executed to deactivate one or more client computers, servers, or network equipment on the local system. Such actions may be performed with the intention of gaining unauthorized access to target devices and disrupting or disabling the applications running on them. In most cases, these attacks are carried out by malicious individuals or cybercriminals. By exploiting vulnerabilities in local network security, attackers can accomplish their objectives with the assistance of viruses, malware and spyware. Such attacks that disable network devices can cause significant material and ethical damage to corporate businesses, including theft of sensitive information[1].

2.1.1 Aims of the attackers

The objective of computer network attacks acquired in accordance with research conducted by the International Cisco Network Academy is outlined below (Cisco Network Academy, 2023):

Information Theft: The act of unlawfully acquiring data from a storage area can be accomplished by gaining entry to a local network or a client user with the assistance of practices such as malware, phishing (Gupta et al., 2016) or brute force (Tams et al., 2015).

Identity Theft: Identity theft involves attackers seizing personal information, including personal identity information, credit card details and online transaction passwords. Attackers can cause material and moral damage to individuals or institutions through the capture of this important data.

Service interruption : Service interruption refers to the disabling of client computers connected to local networks from receiving services from network systems, servers, or related web pages, resulting in system malfunctions. Online services provided by governments and banking transactions have become a serious threat to corporate companies. This threat manifests in forms such as Denial of Service Attacks (DoS) (Sharafaldin et al., 2018) or Distributed Denial of Service Attacks (DDoS) (Akgun et al., 2022).

2.2 Attack Types

2.2.1 Denial of Service Attack

A denial of service attack (DOS) refers to any type of attack carried out on a network structure with the aim of incapacitating a server's ability to serve its clients. Precise identification of the source of the attack is difficult due to the use of a spoofed IP address. Attacks range from sending millions of requests to a server in an attempt to slow it down, flooding a server with large packets of invalid data, to sending requests with an invalid or spoofed IP address[2].

2.2.2 Distributed Deniel of Service Attack

A denial of service attack is identified by an attacker's attempt to prevent authorized users of a service from accessing the required resources. Several examples of denial of service attacks are available[4]:

- Attempts to flood a network in order to prevent legitimate network traffic.
- Attempts to disrupt the connections between two machines in order to prevent access to a service.
- Attempts to prevent a particular person from accessing a service.
- Attempts to disrupt service to a specific system or person.

The distributed format adds a dimension of "many to one", which renders these attacks more challenging to prevent [5]. A distributed denial of service (DDoS) attack comprises four components. The first element is the victim, which refers to the targeted host that endures the attack. The second element concerns the presence of the attack daemon agents, which are software programs responsible for executing the attack on the victim host. Typically, these daemons are deployed in host computers and adversely

impact both the target victim and the host machine. Technical abbreviations are clarified upon first use. Typically, these daemons are deployed in host computers and adversely impact both the target victim and the host machine. Deploying attack daemons requires the attacker to infiltrate host computers. The third component of a distributed denial of service attack is the control master program, which coordinates the attack. The mastermind behind the attack is the actual attacker, who can remain anonymous while using the control master program to orchestrate the attack. [3]

The process of a distributed attack involves the following steps [6]:

1. The attacker sends an "execute" message to the control master program.

2. The control master program receives the "execute" message and disseminates the command to the attack daemons under its jurisdiction.

3. Upon receiving the attack command, the attack daemons initiate the attack on the victim.

While it may appear as though the actual attacker has little involvement beyond issuing the "execute" command, in reality, they must meticulously plan the implementation of a successful distributed denial of service attack. The attacker is required to infiltrate all of the host computers and networks in which the daemon attackers will be deployed, as well as analyse the target's network topology to identify any bottlenecks or vulnerabilities that can be exploited during the attack. Due to the implementation of attack daemons and control master programs, the actual attacker is not directly implicated in the attack, resulting in increased difficulty identifying the perpetrator[3].

2.2.3 Botnet Attacks

Botnets can be used for legitimate as well as illegitimate reasons [7]. One legitimate use is to help run IRC channels by giving certain people administrative privileges. However, this purpose does not explain the large number of bots observed. According to data gathered from Honeypots [8], we can categorize the potential uses of botnets for criminal or destructive purposes as follows. [15]

DDoS Threats: Botnets are frequently used for DDoS threats [8]. These threats disable the network functions of the target system through the consumption of the bandwidth of the affected system. For example, an attacker could instruct the botnet to connect to a victim's IRC channel. The target could be flooded with an overwhelming number of server queries from the botnet. The victim's IRC network is disabled as a

result of this DDoS attack. Evidence shows that threats of UDP and TCP SYN flooding [9] are the most frequently employed by botnets.

General countermeasures against DDoS attacks;

1. Management of large numbers of devices at risk.

2. Disable the remote control system [9].

However, there is still a need for more efficient methods to prevent this type of attack. [15]

Spamming and Spreading Malware: Botnets are responsible for approximately 70% to 90% of the world's spam, causing concern among professionals in the internet protection sector [10, 11]. The study report suggests that compromised hosts may be used for malicious purposes, such as spamming, once the SOCKS v4/v5 proxy (TCP/IP RFC 1928) is opened by certain bots. Additionally, certain bots have the ability to collect email addresses through specific functions [8]. A botnet can be utilized by attackers to send large amounts of spam due to its ability to send spam in huge quantities. [12, 15].

In [13], Trinity, a distributed spam classification system, was proposed by researchers. It is content-independent and designed to combat botnet spamming. The system is based on the assumption that spam bots do a large number of emails in a short period of time. This means that all emails from this source are considered spam. It is unclear how effective Trinity is since it is still being experimented on. [15]

To uncover the collective behaviours of spamming botnets and promote their identification in the future, Xie et al. [14] formulated a spam signature generation framework named AutoRE. The study discovered several identifying features of spamming botnets. Firstly, spammers tend to add random and legitimate URLs to their messages in order to avoid detection [14]. Secondly, the IP addresses of botnets are usually distributed across multiple ASes (Autonomous Systems), with only a small number of machines in each AS on average [14]. Lastly, despite variations in content, spam emails frequently share similar recipient addresses [14]. Researching how to utilise these features to capture botnets and prevent spamming is worth considering for the future. [15]

Botnets can also be used for spreading malware [8]. For example, a botnet could deploy the Witty worm to target the ICQ protocol because the victim's system might not have activated Internet Security Systems (ISS) services [8, 15].

Click Fraud: With the assistance of botnets, perpetrators can install advertising add-ons and browser helper objects (BHOs) for business purposes [8]. This resembles Google's AdSense programme as they seek to achieve higher results. Click-through rate (CTR) can be artificially promoted by perpetrators who use botnets to click on specific hyperlinks periodically [8]. This method also proves effective in online polls or games [8]. Due to each victim's host possessing a distinct IP address distributed worldwide, each click is perceived as a valid action by a legitimate individual. [15]

Identity Fraud: Identity Fraud, also known as Identity Theft, is a rapidly increasing cybercrime [8]. Phishing emails are a common occurrence. The message typically contains URLs that appear to be legitimate and requests that the recipient provide personal or confidential information. This type of email can be generated and sent by botnets using the mechanisms of spamming. [8]. In addition, botnets can also create multiple fraudulent websites that mimic legitimate businesses in order to obtain personal information from unsuspecting victims. A new fake website may appear until the computer is shut down after a fake website has been closed by its owner. [15]

2.2.4 Brute Force Attacks

The process of guessing passwords by attempting multiple combinations, otherwise known as a Brute force attack, is executed through the "Trial and error" method. An attacker initially collects essential information about the user. For example, user’s full name, room number, vehicle number, children names etc. The attacker persistently attempts passwords randomly using the user's personal information. It is possible that this can take from hours, days, months, or even years. The more advanced the encryption scheme (32, 64, 128, 168-bit, etc.) utilized, the greater the length of time necessary. The greater the level of encryption applied (32, 64, 128, 168 bit etc.), the longer it takes to complete the process. [16]

As indicated in the table above, it is possible to make alterations depending on the size of the key. The longer the key, the less likely it is to be compromised. A brute force attack is also referred to as a "Dictionary attack" or a "Hybrid Brute-force attack".

If your website demands user authentication, it becomes a prime target for brute-force attacks. Such an attack aims to accurately deduce your password by exhaustively trying every imaginable combination of letters, numbers, symbols, etc. However, the drawback is that it can take a considerable amount of time (i.e., years) contingent upon the password's complexity and length. To avoid this scenario, numerous individuals

prefer to select a dictionary word that holds meaning to them, instead of opting for a random password. A "Dictionary attack" is executed by the attacker using exact dictionary words, whereas a "Hybrid Brute-force attack" is performed by slightly modifying the dictionary words. Attackers have access to various tools that can generate numerous possible combinations during an attack, with each try originating from a different IP address, making it difficult to track a single account for unsuccessful password attacks. [16]

2.2.5 Port Scanning Attacks

Port scanning and Distributed Denial-of-Service (DDoS) attacks are frequently employed by cyber attackers to identify vulnerabilities and overwhelm a target's resources. It is imperative to implement measures to prevent and mitigate these attacks to ensure the security of information systems. During the port scanning process, a scanning tool detects open ports on the target and reports the running services, as well as provides an enumeration of the target's status, such as the operating system in use, the amount of memory occupied, and processing information. The objective of port scanning is to detect the vulnerable areas of a target resource, which might be exploited.

According to Oracle® [1], a typical attack follows five steps: reconnaissance, enumeration, penetration, exfiltration, and sanitation.

After collecting adequate data such as IP scheme, data center locations, and target profile during the reconnaissance phase, port scanning is performed as an initial stage of the enumeration process. NMAP is widely used as the most prevalent tool for port scanning globally [18]. A typical use case involves scanning for ports that permit Transmission Control Protocol / Internet Protocol (TCP/IP) traffic. To scan a port, a SYN signal is dispatched to the target by the scanner. If the target responds with a SYN+ACK signal, the port is explicitly open. In order to terminate the connection, the following step is to close it. If it is determined that the port is open, the scanner may send a reset (RST) signal. Alternatively, leaving the connection open could lead to a denial of service attack known as a TCP SYN attack. [17]

2.2.6 Web Application Attacks

Web attacks pose a significant threat to web applications as they can potentially compromise user data and application source code. The different classifications of web

attacks have been categorised according to their impact on victims or the approach they use to attack an application. Understanding vulnerabilities is the first step towards removing them. The following section aims to aid in the identification and grouping of attacks. Six different attack types are presented, each comprised of several specific attacks. There are six categories of attacking methods: Command Execution, Information Disclosure, Logical Attacks, Authentic Based Attacks, Authorization Based Attacks, and Client-Based Attacks [19, 23].

2.2.6.1 Authentication-Based Attacks

There are several Autentication-Based Attacks

2.2.6.1.1 Brute Force Attack

This method is automated and used to determine an unknown value by testing a large set of possible values. Successful arrival at the correct value can allow access to private and sensitive information [20].

2.2.6.1.2 Insufficient Authentication

If a web service or web application grants access to its content or functionality without proper authentication, attackers may easily exploit this vulnerability [20].

2.2.6.1.3 Weaknesses in Password Recovery

A flawed password recovery system can enable an attacker to acquire, alter, or retrieve a user's password. Such weaknesses may be exploited through brute force methods, guessing the correct answer to a security question, or taking advantage of inherent weaknesses in the system. [23]

2.2.6.2 Authorization Based Attacks

There are several Authorization Based Attacks

2.2.6.2.1 Session Predication/Session Hijacking

As implied by its name, this attack type involves the hijacking or impersonation of a user on a website by the attacker. The main concern in this scenario is the session ID,

which, if predicted by the attacker, poses a significant risk, engage in fraudulent activity. [23]

2.2.6.2.2 Insufficient Authorization

A website's security policy is vulnerable if there is no means of verifying whether a user adheres to it when performing actions [20].

2.2.6.2.3 Insufficient Session Expiry

Insufficient session expiry arises when a web application enables an attacker to reuse an outdated session ID to gain authorization. [23]

2.2.6.2.4 Session Fixation

This attack technique involves the forced assignment of an explicit value to a user's session ID. [23]

2.2.6.3 Client Side Attacks

2.2.6.3.1 Content Spoofing

Content Spoofing refers to an attack where an attacker inserts potentially harmful material into a website, which the user may view as genuine [20].

2.2.6.3.2 Cross-Site Scripting

Cross-site scripting (XSS) is a web-based attack that involves injecting untrusted data into a website that is perceived to be trustworthy. The compromised website is then utilized to store, transport, or distribute malicious content to the victim [21].

2.2.6.4 Command Execution Attacks

There are several Command Execution Attacks

2.2.6.4.1 Buffer Overflow

A Buffer Overflow attack happens when too much information is written to a memory block, also called a buffer, causing it to exceed its capacity. [23]

2.2.6.4.2 Format String Attack

Format string attacks utilize string formatting library features to access another memory space, and may also modify the flow of a web application, posing a security risk. [23]

2.2.6.4.3 Light Weight Directory Access Protocol (LDAP) Injection

Applications that construct LDAP statements from user input are vulnerable to LDAP Injection attacks, causing significant concern.

2.2.6.4.4 Operating System (OS) Commanding

OS command injection is an attack in which an attacker can execute unauthorised operating system commands via user input manipulation on a web application.

2.2.6.4.5 SQL Injection

SQL injection is a frequent attack in applications that have the ability to generate SQL queries from user input. If successful, the attacker can gain access to the application database and manipulate it using SQL statements [22].

2.2.6.4.6 Server-Side Include (SSI) Injection

In Server-Side Include injection, the attacker can infiltrate a web application by inserting code that will be executed by the server later [21].

2.2.6.4.7 X-Path Injection

In X-Path Injection Attacks, applications that employ user-supplied input to construct XML documents to query XML are highly vulnerable to web attacks. [23]

2.2.6.5 Information Disclosure

There are several instances of information disclosure.

2.2.6.5.1 Directory Indexing

Directory Indexing is a type of server function that discloses all the results within the requested directory if the regular base file is absent [20, 23].

2.2.6.5.2 Information Leakage

Information leakage is a security weakness whereby an application discloses specific technical information about itself, as well as private and sensitive data, for example, user details. [23]

2.2.6.5.3 Path Traversal

A path traversal attack enables attackers to access files and documents located outside of the base directory. [23]

2.2.6.5.4 Predictable Resource Location

In this type of attack, the attacker is able to access hidden website functionalities and content. [23]

2.2.6.6 Logical Attacks

There are several Logical Attacks.

2.2.6.6.1 Abuse of Functionality

This refers to a type of attack in which a web application uses its features and functionalities to target itself or others [20, 23]. Web applications can be vulnerable to self-inflicted attacks where the application's features and functions are used to attack itself or others [20].

2.2.6.6.2 Denial of Service

Denial of Service (DoS) attacks prevent an application from functioning according to user desires, but rather align with the attacker's wishes. [23]

2.2.6.6.3 Insufficient Anti-Automation

Insufficient Anti-Automation refers to a type of attack where an attacker is given permission to automate a process that was intended to be executed manually. [23]

Chapter 3

Materials and Methods

3.1 Machine Learning

Machine learning is a broad discipline encompassing information technology, statistics, probability, artificial intelligence, psychology, neurobiology, and various other fields. A good representation of a chosen dataset can be achieved through constructing a model. Machine learning is a simple solution for problem-solving. Machine learning has become a sophisticated field, teaching computers to mimic the human brain. It has broadened the discipline of statistics, generating essential statistical and computational theories of the learning processes.[24]

Machine learning involves the creation of algorithms that enable computers to learn autonomously. By detecting statistical regularities or other patterns in data, individuals can learn. The machine learning algorithms [25] are modelled on human learning approaches. Additionally, these algorithms can provide insight into the relative complexity of learning in various settings. [24]

These days, the development of new computing technologies in the field of Big Data and machine learning differs from past practices. At present, many machine learning algorithms have been developed [26], updated, and refined. The latest advancement in machine learning is the ability to autonomously apply intricate mathematical calculations to large datasets, which results in faster computation. [24]

3.1.1 Supervised Machine Learning

Supervised learning typically leaves input probability undefined, even when the expected output is known. Thus, a dataset results that contains features and labels. The primary objective is to create an estimator that can forecast an object's label from a set of features. Thereafter, the learning algorithm takes in a set of features as input while also receiving the correct outputs. It then learns through comparing its actual output to

the corrected outputs to identify errors. The model is adjusted correspondingly, based on the inputs provided. Should any input values be missing, it becomes impossible to deduce any information regarding the outputs. There is no requirement for the creation of the model as long as the inputs are accessible. [24]

3.1.2 Classification

3.1.2.1 K-NN (K-Nearest Neighbours)

The K-Nearest Neighbors (KNN) algorithm is a nonparametric classification method, meaning it refrains from making assumptions about the input data. It is renowned for its efficacy and straightforwardness and operates as a supervised learning model. The algorithm learns from labeled training data. A dataset is available with classified data points that permit the prediction of the class of unlabeled data.

In classification, the unlabeled data's class is determined by various characteristics. KNN is primarily employed as a classifier, classifying data based on its closest proximity or. neighboring training examples in a given region. This method is used for its simplicity of execution and low computation time. For continuous data, it uses the Euclidean distance to calculate its nearest neighbors.

For a new input, the K-nearest-neighbours algorithm calculates the closest neighbours and determines the classification for the new input based on the majority decision of these neighbours. Although this classifier is straightforward, the value of 'K' significantly impacts the classification of the unlabeled data. [28]

3.1.2.2 SVM (Support Vector Machine)

Support Vector Machines (SVMs) are a recent learning approach employed for binary classification. The fundamental concept is to identify a hyperplane that perfectly segregates the d-dimensional data into its two classes. Nevertheless, since sample data is frequently not linearly separable, SVMs introduce the concept of kernel tricks. notion of a "kernel induced feature space" which casts the data into a higher dimensional space where the data is separable. Typically, casting into such a space would cause problems computationally, and with overfitting. The key The insight employed in support vector machines is that dealing with the higher-dimensional space directly is unnecessary; the dot-product formula for that space suffices, thus eliminating the aforementioned concerns. Additionally, the VC-dimension (a measure of the likelihood of a system to

perform optimally) is taken into consideration. Unseen data of SVMs can be calculated explicitly, unlike other learning methods such as neural networks, for which there is no measure. Overall, SVM's are intuitive, theoretically well- founded, and have shown to be practically successful. SVM's have also been extended to solve regression tasks (where the system is trained to output a numerical value, rather than "yes/no" classification). [30]

3.1.2.2 Random Forests

The Random Forest algorithm was developed by combining the BAGGing method (Breiman, 1996) with random variable selection to create a single "strong learner" from a group of "weak learners". Each decision tree within the group is created using a sample from the training data with replacement. These decision trees act as base estimators for establishing class labels of unlabeled instances through majority voting. Abbreviations for technical terms are explained upon first use. Bootstrap sampling is employed for every tree in the Random Forest, with data being split based on whether the problem is for Classification or Regression. When it comes to classification, the Gini index is employed to partition the data, whilst for regression, minimizing the sum of the squares of the mean deviations can be opted for when training each tree model. The usage of Random Forests provides advantages such as employing ensembles of trees without any pruning. [29]

3.1.2.2 XGBoost Classifier

XGBoost is short for eXtreme Gradient Boosting (Chen & Guestrin, 2016). It is a type of Gradient Boosting Machine (GBM) that is used extensively for creating predictive models in both classification and regression problems. It has been observed that XGBoost significantly outperforms other machine learning algorithms, as demonstrated in various machine learning datasets (Friedman, 2001). XGBoost is an ensemble technique where novel models are created to fix the residuals or errors of earlier models before being merged to provide the ultimate prediction. Empirically, XGBoost displays relatively faster performance when compared to various other ensemble classifiers, such as AdaBoost. The XGBoost algorithm has gained widespread recognition in various machine learning and data mining challenges. It has become a popular tool among the competitors of Kaggle and data scientists in the industry

(Nielsen, 2016). Furthermore, it is a parallelizable algorithm, enabling the utilization of the multi-core computers, thus facilitating the training of large datasets. Moreover, XGBoost is freely available open source software which can be used under the permissive Apache-2 license.1[29]

3.1.2.2 LightGBM (Light Gradient Boosting Machine)

LightGBM is a gradient-based learning framework that leverages decision trees and boosting concepts (Ke et al., 2017). This model is relatively new and will be described in detail here. Unlike the XGBoost model, its primary distinguishing feature is the use of histogram-based algorithms to accelerate training, reduce memory consumption, and implement a leafwise growth strategy with depth constraints. The fundamental concept behind the histogram algorithm is to categorize uninterrupted eigenvalues into k divisions and generate a histogram that has a k-width. Furthermore, it allows the retention of results after the feature discretization, which typically needs only an 8-bit integer, thus lowering the memory consumption to 1/8 of the original amount. This algorithm does not necessitate additional storage for pre-sorted findings. In addition, the algorithm's crude partitioning approach does not decrease the accuracy of the model.

Since decision trees are considered weak models for studying, it is not crucial whether the segmentation point is accurate or not. More general segmentation points can have a regularization effect, which can prevent overfitting effectively. The growth strategy for decision trees is referred to as. Leaf-wise is a more effective strategy than level-wise, as it treats each leaf individually instead of grouping them by layer. This results in lower memory consumption as only the leaves with the highest branching gain are considered at each step. After completing the branching cycle, the blade can reduce more errors and achieve higher precision compared to the horizontal direction, with the same number of segmentation efforts. However, using a leaf orientation may lead to the growth of deeper decision trees and result in overfitting. Therefore, to ensure high efficiency and prevent overfitting, LightGBM implements a maximum depth limit at the top of the leaf. [29].

3.1.3 Hyperparameter Optimisation

Machine learning models are fundamentally founded on mathematical functions which illustrate the relationship between dependent and independent variables. The process of hyperparameter optimisation involves discovering the optimal combination of hyperparameters for a machine learning algorithm based on a predetermined measure of success. By optimising the model complexity with hyperparameter optimisation, the issues of overfitting and underfitting can be counterbalanced. Again, the issue of overfitting resulting from the model's flexibility can be resolved by imposing constraints through hyperparameters.

With numerous hyperparameters and values available for a machine learning algorithm, it is apparent that it would be challenging to manually test each combination and select the optimal one. As a result, various methods have been developed for hyperparameter optimisation, including GridSearchCV and RandomisedSearchCV.

3.1.3.1 GridSearchCV

To evaluate the hyperparameters and their corresponding values in the model, the model is constructed individually with each combination and subsequently, the hyperparameter set that yields the most favourable outcomes according to the designated metric is selected. [33]

3.1.3.2 RandomisedSearchCV

A set of hyperparameters are selected at random and the model is assessed through cross-validation. This process proceeds until either the computation time limit or the designated number of iterations is achieved. [34]

3.1.3.3 BayesSearchCV

BayesSearchCV incorporates a "fit" and a "score" technique. Additionally, if they are present in the estimator utilised, it incorporates "predict", "predict_proba", "decision_function", "transform" and "inverse_transform". Cross-validated searches are used to refine the parameter settings of the predictor used to apply these techniques. Unlike GridSearchCV, it does not try all parameter values, but extracts a fixed number of parameter sets from the given probabilities. The n_iter denotes the number of attempts to set the parameters. [35]

3.2 Python Programming Language

Python is an object-oriented, interpretive programming language that is widely acclaimed for its versatility in handling a diverse range of assignments (Helmus & Collis, 2016). Since its inception, the open-source nature of this has greatly increased its importance. Python was created in 1991 by Van Rossum and Drake Jr (1995) and is now considered one of the most important programming languages to study (Saabith, Fareez, & Vinothraj, 2019).

It is open source, compatible with all platforms across Windows, Mac, Ubuntu and Linux, and has low system requirements, so anyone can write code in Python. Furthermore, there is already a vast community spanning various disciplines and skill levels. From ordinary people to leading researchers, individuals have created fascinating data science projects, machine learning, artificial intelligence, app and game development, as well as scientific research and more. These projects showcase a wide range of innovative solutions and promising possibilities for the future. Python resources are readily available thanks to the open-source community that continually improves its language capabilities. The community also offers extensive resources, which can be accessed simply by adding the keyword "Python" to any search. Such resources include courses, source code, solutions to frequently encountered issues, video instructions, and additional resources. These materials are frequently available at no cost and address common issues encountered by developers at varying levels of difficulty.

By using this approach, you can save time, increase experimental control, and let your creativity guide your research designs, ultimately optimizing your research efficiency. With these targets in mind, this manual has been designed to help you start your research on a solid footing. As quickly as possible, this will show you how to use the Anaconda deployment toolset to install the packages in the most efficient way. The guide will cover the usage of these packages, essential terminology, various Python interfaces, and implementing your initial few lines of code to create a functional program. This task aims to prepare you for the next few steps in learning Python.

Using a research management tool can save time, increase experimental control, and optimize productivity. It also allows for more creative research designs. [31]

3.2.1 Anaconda

Anaconda is a software that can be freely downloaded and is designed for scientific research purposes. By installing Anaconda, you will have access to several environments which can be used for coding in both Python and R. These environments, known as integrated development environments (IDEs), are useful platforms that simplify the code development process. IDEs play a comparable function to text processors such as Microsoft Word, Google Docs, and Pages for text composition, however, they offer much more. These Integrated Development Environments are packed with a multitude of helpful features for composing, modifying, and debugging code, inspecting as well as visualizing data, storing variables, presenting outcomes, and collaborating on projects. Although IDEs can vary in appearance and complexity, the underlying programming language remains the same. Therefore, changing IDEs within Anaconda does not significantly alter the Python in the code. The learning curve for understanding Python's syntax is a significant factor. Once you have learned how to code in one IDE, you will be able to transfer this skill to another with ease. No single IDE is objectively better than another, as each may have its own advantages and disadvantages. The choice of IDE should be based on personal preference. [31]

3.2.2 Anaconda Navigator and IDEs

As previously stated, the Anaconda Navigator is a graphical user interface (GUI). The website serves as the homepage for Anaconda and provides easy access to various Python programming integrated development environments (IDEs). This negates the need to utilise terminal commands to launch the IDEs. This is a set of IDEs that are specifically designed for Python programming. They can be installed using Anaconda. To start an IDE, simply click on its icon located in the Navigator main window. [31]

3.2.3 Jupyter Notebook: The Classic Notebook Interface

Jupyter Notebook is a web-based integrated development environment that utilizes your default web browser. Each block of code can be executed separately, enabling great flexibility and experimentation. This feature permits the use of various kinds of text in one Notebook, including code outputs, visualisations, equations, and plain text. This is an effective tool for creating and sharing documents, presenting code and results in an organised and attractive manner. As it is web-based, it facilitates sharing Notebooks with others and is optimal for collaborations. [31]

3.2 VHS-22 Dataset

The source data is a fusion of multiple datasets, including CTU-13, CICIDS 2017, ISOT, Booters, and traffic examples from MTA. This selection is based on verifiable facts. All datasets and network traffic patterns are marked, freely available and source independent. [32]

The CICIDS 2017 dataset contains network traffic that was generated and collected in a simulated university network. Therefore, it provides a reliable representation of traffic in real networks. [32]

The CICIDS 2017 dataset categorises attacks by threat type, including the most commonly reported ones. [8]. The following are examples of cyber threats: DoS and DDoS attacks, web attacks, web scanning, brute force attacks, botnet traffic, infiltration, and exploitation of the Heartbleed vulnerability. This dataset provides a comprehensive representation of real-world cyber threats. [32]

The traffic on the MTA website is current and generated by genuine malicious samples. Each sample in Booters contains over 100 attack records, providing typical examples of DDoS threats. [32]

CTU-13 includes non-peer-to-peer traffic, which is uncommon among other datasets. [32]

The ISOT dataset is a composite of four other datasets and includes typical traffic produced by various everyday applications, including games, among other things. This provides additional value. [32]

Therefore, it is believed that the dataset resulting from diverse sources will be highly heterogeneous, posing a challenge in the design of intrusion detection algorithms. [32]

The dataset consists of various attack types such as botnet, malware, web attacks, brute-force, DDoS, and DDoS Booter attacks. [32]

Table 3.1 The VHS-22 provides access to extracted stream parameters, including descriptions of flow level and network level properties.

	Feature	**Description**
1	ip_src_str	IP address of the source
2	ip_dst_str	IP address of the destination
3	ip_protocol	This document outlines the requirements

		for addressing and routing data on the Internet
4	sport	This is a number that indicates the port to which the source IP address is connecting
5	dport	This is a number that indicates the port to which the destination IP address is connecting
6	in_packets	The total number of packages in the stream
7	b_packet_total	The total number of bytes in the stream
8	first_timestafirst	The value of timestamp in the stream
9	last_timestamp	The value of last timestamp in the stream
10	duration	Flow time
11	flags_sum	The total number of TCP flags
12	urg_nr_count	The total number of URG flag in the stream
13	ack_nr_count	The total number of ACK flag in the stream
14	rst_nr_count	The total number of RST flag in the stream
15	fin_nr_count	The total number of SYN flag in the stream
16	psh_nr_count	The total number of PSH flag in the stream
17	syn_nr_count	The total number of SYN flag in the stream
18	b_packet_max	size value of the biggest package
19	b_packet_min	size value of the minimum package
20	b_packet_mean	size value the average package
21	b_packet_median	The median value of the package
22	b_packet_first_q	First quartile value of package
23	b_packet_third_q	The 75th percentile value of packages
24	b_packet_std	The standard deviation of the package is

		as follows
25	b_packet_total	The total size value of flow in bytes
26	iat_min	The least inter-arrival times
27	iat_max	value of the top inter-arrival times
28	iat_first_q	First quartile value of inter-arrival times
29	iat_third_q	The 75th percentile value of inter-arrival times
30	iat_std	The standard deviation of inter-arrival times
31	iat_mean	value of the mean of inter-arrival times
32	iat_median	value of the median of inter-arrival times
33	iat_var	value of the variance of inter-arrival times
34	connections_from_this_host	Inbound traffic from the specified host
35	connections_to_this_host	Number of connections to the host machine
36	connections_rst_to_this_host	Number of connections to host ended with RST flag
37	connections_rst_from_this_host	Inbound traffic from the host ended with RST flag
38	connections_to_this_port	Number of traffic with the same destination port
39	connections_from_this_port	Number of traffic with the same source port
40	connections_ratio_from_this_host	The percentage of connections to the host with the same destination address.
41	connections_ratio_to_this_host	The percentage of incoming connections from hosts with the same source address
42	connections_ratio_rst_to_this_host	The percentage of connections that ended with the RST flag sent by the host.
43	connections_ratio_rst_from_this_host	The percentage of incoming connections from the host computer that ended with the RST flag
44	connections_ratio_to_this_port	The percentage of connections to the host

		with the same destination port
45	connections_ratio_from_this_port	The percentage of incoming connections from the host with the same source port

Chapter 4

Results

4.1 Sample Dataset

The VHS-22 dataset is a heterogeneous collection at the stream level, which consolidates the datasets of ISOT, CICIDS-17, Booters, and CTU-13, in addition to traffic gathered from the Malware Traffic Analysis (MTA) site. The dataset promotes enhanced diversity in both legitimate and illegitimate traffic flows. It features 27.7 million flows, comprising 20.3 million legitimate flows and 7.4 million attack flows. The flows are represented in the form of 45 features.

As the dataset was excessively large, we created a new sample comprising 5.5 million traffic flows consisting of 27% attacks and 73% normal network traffic flows. Following this, we carried out experiments on this sample. The sample network includes 110 unique categories of attack traffic.

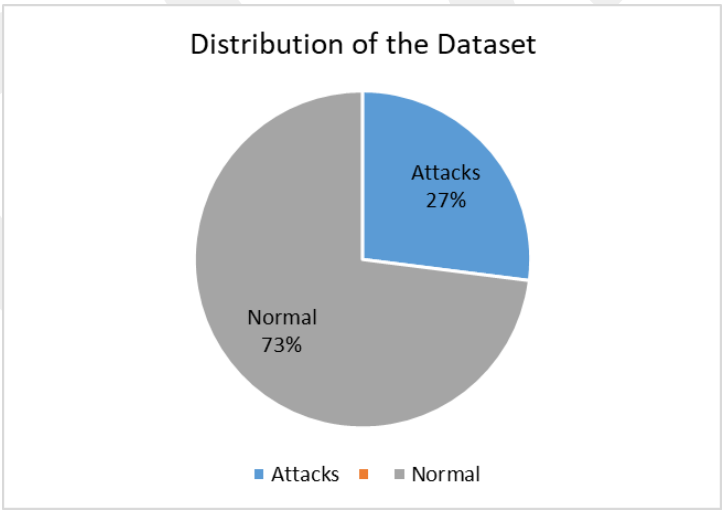

Figure 4.1 Distribution of sample dataset

We conducted our work in the Jupyter Notebook web application due to its user-friendly, streamlined, and document-centric design. Some data blocks were presented as

objects, and to address this, we implemented the LabelEncoder Library on Jupyter Notebook to transform the object values into numeric values.

By using Standardscaler, the dataset was adjusted to have a mean of zero and a standard deviation of one for every attribute value. This process helps the learning algorithms perform optimally, especially when the dataset has characteristics at diverse scales.

With the assistance of "VarianceThreshold(threshold=(.8 * (1 - .8)))", low variance features were identified and eliminated from the dataset. The following features were removed:

- iat_first_q,
- iat_third_q,
- urg_nr_count,
- connections_rst_from_this_host,
- connections_ratio_from_this_host,
- connections_ratio_to_this_host,
- connections_ratio_rst_from_this_host,
- connections_ratio_to_this_port.

Then, we deleted the label and attack_file columns from dataset.

We utilized the StratifiedGroupKFold function in Python to partition the sample dataset into training and test datasets. This function executes StratifiedGroupKFold cross-validation on a Python dataset comprising two group labels. The cross-validation technique, StratifiedGroupKFold, partitions the data into train and test sets, maintaining the distribution of the target variable and the groups. This function requires the input of features, the target variable, and group labels for each sample, and yields a generator object generating train and test indices for each fold. The function allows for the specification of the number of folds as an optional argument, utilizing the StratifiedGroupKFold implementation as provided by the scikit-learn library. For this purpose, we combined the ip_src_str and ip_dst_str columns, resulting in a new column named ip_str (data["ip_str"] = data["ip_src_str"] + '_' + data["ip_dst_str"]). In the case of splitting, such as the training and test dataframes, we used this ip_str variable that we created. After parsing the train and test sets, we removed the columns for ip_str, ip_src_str and ip_dst_str from both datasets.

A clustering analysis was conducted on the dataset to separate the information into groups according to certain proximity criteria. The K-Means assignment mechanism, one of the most widely used unsupervised learning methods, was employed during the clustering analysis, as it allows each data point to belong to only one cluster [36]. This resulted in the creation of a new cluster feature on the dataset.

4.2 Random Forests Classifier

Random Forests Classifier is a machine learning algorithm that is widely employed, and it is trademarked by Leo Breiman and Adele Cutler. The algorithm amalgamates the outcomes of numerous decision trees to produce a singular outcome. we felt the need to use it because of its ease of use and flexibility, and because it addresses both classification and regression problems. In this model, we have chosen the n_estimators attribute and assigned it a value of 100, which represents the number of trees we wish to generate before taking the maximum votes or prediction averages. Balanced class weights can be automatically calculated with the sample weight function. To adjust the weights inversely proportional to the class frequencies in the input data, we have set class_weight='balanced'. We have attained 99% success in multi-class classification. Given the remarkable success rate, we investigated whether it could be attributed to a feature associated with the Attack label. We have identified the most significant features using the feature_importances_ attribute in Random Forests Classifier.

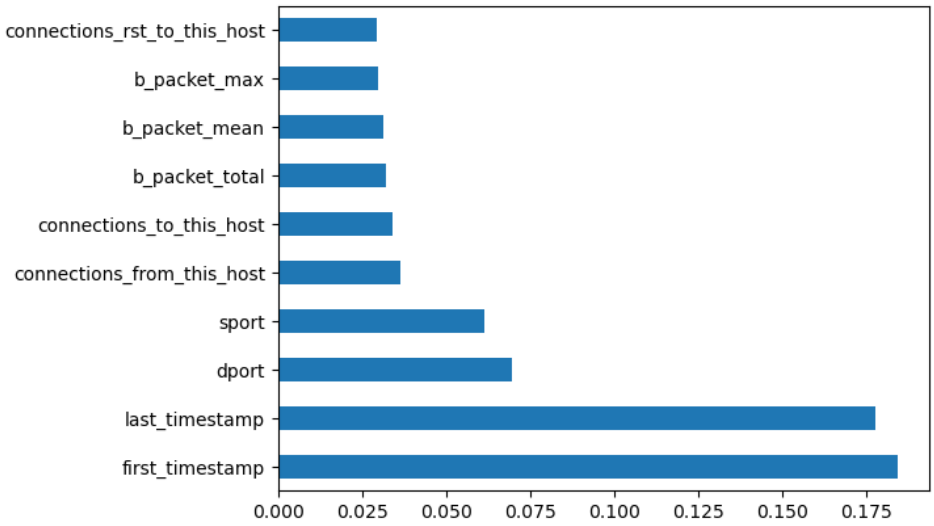

Figure 4.2.1 Feature importances associated with Attach Label feature.

We discovered the top 5 significant features (last_timestamp, first_timestamp, ip_str, dport, sport) and implemented the model again after eliminating them from the dataset. This approach produced a more accurate success rate of 90.70%.

Table 4.2 The scores obtained by applying the Random Forest model are presented below.

RandomForestClassifier	Values
Dataset	Sample Dataset- %20 VHS-22 Dataset
Accuracy Score	0.9070
Cross_Validation_Score	[0.9871, 0.9872, 0.9871, 0.9873, 0.9872]
f1_score	0.9402
precision_score	0.9788
recall_score	0.9070
AUC	0.6195

4.3 XGBoost Classifier

XGBoost is a machine learning system that combines gradient boosting and decision tree-based methods. We initially applied this model to the datasets that we acquired.

- XGBClassifier(objective="multiclass:softmax", num_class=110)

In this study, a multiclass objective was selected, and it was specified that there were 113 categories. This approach produced success rate of 73.59%.

Table 4.3 The scores obtained by applying the XGBoost Classifier model are presented below.

XGBoost Classifier	Values
Dataset	Sample Dataset- %20 VHS-22 Dataset
Accuracy Score	0.7359
Cross_Validation_Score	[0.8584, 0.8539, 0.9165, 0.8483, 0.9301]
f1_score	0.7347
precision_score	0.7342
recall_score	0.7359
AUC	0.5032

4.4 K-NN Classifier

In K-nearest neighbor (K-NN) classification, a class membership is determined by a majority vote of the nearest neighbors assigned to an object. The object is subsequently assigned to the class that is most frequent amongst its k-nearest neighbors. By default, I have selected 5 neighbors to be used for K-NN queries. With this method, we attained a 73.% success rate.

Table 4.4 The scores obtained by applying the K-NN Classifier model are presented below.

K-Nearest Neighbors Classifier	Values
Dataset	Sample Dataset- %20 VHS-22 Dataset
Accuracy Score	0.9739
Cross_Validation_Score	[0.9850, 0.9862, 0.9859, 0.9863, 0.9859]
f1_score	0.9722
precision_score	0.9724
recall_score	0.9739
AUC	0.5345

4.5 SVM

The algorithm is designed for solving classification problems, utilizing a more flexible representation of class boundaries and automatic complexity control. It is capable of finding a single global minimum in polynomial time. Our successful application of this approach yielded a 91.09 % success rate.

Table 4.5 The scores obtained by applying the Random Forest model are presented below.

SVM	Values
Dataset	Sample Dataset- %2 VHS-22 Dataset
Accuracy Score	0.9109
Cross_Validation_Score	[0.9102, 0.9095, 0.9099, 0.9094, 0.9118]
f1_score	0.9013
precision_score	0.9077
recall_score	0.9109
AUC	0.5902

4.6 LightGBM Classifier

LightGBM is a gradient boosting framework that uses tree based learning algorithms. We felt the need to use this model due to its faster training speed and higher efficiency, lower memory usage, better accuracy, support for parallel, distributed and GPU learning, and its ability to process large-scale data. We implemented it with attributes values objective="multiclass", num_class=110, n_estimators=100. We achieved 87.83 % success with this method.

Table 4.6 The scores obtained by applying the LightGBM Classifier are presented below.

LightGBMClassifier	Values
Dataset	Sample Dataset- %20 VHS-22 Dataset
Accuracy Score	0.8783
Cross_Validation_Score	[0.7318, 0.5178, 0.5144, 0.8073, 0.6996]
f1_score	0.8653
precision_score	0.8715
recall_score	0.8783
AUC	0.5054

4.7 Hyperparameter Optimisation

The results were obtained by using different parameters for each model and applying GridSearchCV, RandomizedSearchCV, and BayesSearchCV techniques on separate models.

Table 4.7 The scores obtained by applying GridSearchCV, RandomizedSearchCV, and BayesSearchCV techniques are presented below.

Modeller	Accuracy Score	Cross Validation Score	f1 Score	Precision Score	Recall Score	AUC
Random Forest Classifier	0.9070	[0.9871, 0.9872, 0.9871, 0.9873, 0.9872]	0.9402	0.9788	0.9070	0.6195
XGB Classifier	0.7359	[0.8584, 0.8539, 0.9165, 0.8483, 0.9301]	0.7347	0.7342	0.7359	0.5032
LGBM Classifier	0.8783	[0.7318, 0.5178, 0.5144, 0.8073, 0.6996]	0.8653	0.8715	0.8783	0.5054
K-NN Classifier	0.9739	[0.9850, 0.9862, 0.9859, 0.9863, 0.9859]	0.9722	0.9724	0.9739	0.5345
SVM	0.9109	[0.9102, 0.9095, 0.9099, 0.9094, 0.9118]	0.9013	0.9077	0.9109	0.5902
GridSearchCV-XGBClassifier	0.9798	[0.9795 0.9791 0.9785 0.9791 0.9789]	0.9756	0.9734	0.9798	0.9099
GridSearchCV-LGBMClassifier	0.8752	[0.9119 0.8933 0.8654 0.8580 0.9178]	0.9174	0.9667	0.8752	0.2185
GridSearchCV-KNeighborsClassifier	0.9788	[0.9831, 0.9839, 0.9837, 0.9837, 0.9839]	0.9760	0.9742	0.9788	0.5352
GridSearchCV-SVM	0.9374	[0.9527, 0.9524, 0.9514, 0.9511, 0.9515]	0.9638	0.9620	0.9663	0.8492
GridSearchCV-Random Forest Classifier	0.9397	[0.9527, 0.9524, 0.9514, 0.9511, 0.9515]	0.9554	0.9723	0.9397	0.6095
RandomizedSearchCV-XGB Classifier	0.9800	[0.9864, 0.9858, 0.9854, 0.9859, 0.9858]	0.9761	0.9736	0.9800	0.9576
RandomizedSearchCV-LGBM Classifier	0.6923	[0.6764, 0.8511, 0.5124, 0.8111, 0.8169]	0.6491	0.6603	0.6923	0.5005
RandomizedSearchCV-K-NN Classifier	0.9786	[0.9826, 0.9826, 0.9829, 0.9831, 0.9828]	0.9756	0.9736	0.9786	0.5350
RandomizedSearchCV-SVM	0.9663	[0.9240 0.9224, 0.9237, 0.9227, 0.9248]	0.9638	0.9620	0.9663	0.8800
RandomizedSearchCV-Random Forest Classifier	0.9374	[0.9437, 0.9418, 0.9519, 0.9428, 0.9446]	0.9542	0.9724	0.9374	0.6275
BayesSearchCV- XGB Classifier	**0.9811**	**[0.9878, 0.9872, 0.9870, 0.9873, 0.9875]**	**0.9772**	**0.9751**	**0.9811**	**0.9504**
BayesSearchCV-LGBM Classifier	0.8339	[0.8553, 0.7570, 0.8388, 0.7139, 0.7055]	0.8726	0.9178	0.8339	0.5123
BayesSearchCV - K-NN Classifier	0.9788	[0.9861, 0.9855, 0.9854, 0.9858, 0.9851]	0.9759	0.9741	0.9788	0.5351
BayesSearchCV-SVM	0.9663	[0.9709, 0.9724, 0.9708, 0.9715, 0.9712]	0.9638	0.9620	0.9663	0.8995
BayesSearchCV-Random Forest Classifier	0.9399	[0.9412, 0.9425, 0.9500, 0.9423, 0.9460]	0.9555	0.9722	0.9399	0.6288

The highest and most accurate score was obtained from the XGBoost Classifier model with the help of BayesSearchCV technique.

Chapter 5

Conclusions and Future Prospects

5.1 Conclusions

Investigating new methods of identifying network threats, such as those associated with malware, necessitates large and varied datasets. Many network traffic datasets have been suggested and utilised by the research community. Nonetheless, the majority of these datasets are homogeneous and make it relatively easy to detect threats. Consequently, they achieve an accuracy rate of nearly 100%, rendering them no longer challenging. In our research, we have conducted a comprehensive analysis of a diverse range of network traffic data. The dataset was created using a software network probe in combination with pre-existing datasets. We contend that the information contained in the dataset necessitates meticulous scrutiny, thereby affording an outstanding prospect to enhance our proficiency in pinpointing network threats. Our utilization of machine learning algorithms has yielded highly precise results in identifying these threats.

5.2 Societal Impact and Contribution to Global Sustainability

This study aims to further scientific research in the eradication of vulnerabilities in Internet network communication against network attacks. The findings of this study will inform the development of software that can enhance the overall health of Internet network security. It will also provide a basis for future studies to build upon. The safeguarding of sensitive data in public or private institutions will mitigate economic damages and protect corporate reputation. Ensuring the security of company-specific information will prevent financial losses. Confidential information and documents remain secure within the internet-based communication network of national security institutions, bolstering security measures.

5.3 Future Prospects

The model created in this research will have a significant impact on reducing network attacks. However, applying machine learning algorithms to this model may cause increased computational overheads when training the system to identify network attacks. Since new types of network attacks arise frequently, this model may not be able to prevent these novel attacks entirely in the future. It is important to note that the performance of the model largely depends on the accuracy of machine learning algorithms. In order to achieve greater accuracy of results, it could be necessary to employ suitable machine learning algorithms and datasets, and to conduct further studies encompassing new forms of attacks.

BIBLIOGRAPHY

[1] KILINÇ, F., EYÜPOĞLU, C., (2023). Ağ Ortamındaki Saldırı Türleri: Saldırı Senaryo Örnekleri, Journal of Technology and Applied Sciences 6-1 pp99-109 DOI: 10.56809/icujtas.1282687

[2] Elleithy, Khaled M. et al., "Denial of Service Attack Techniques: Analysis, Implementation and Comparison." Journal of Systemics, Cybernetics, and Informatics 3.1, pp. 66-71, 2005.

[3] Lau, F., Rubin, S. H., Smith, M. H., & Trajkovic, L. (2000, October). Distributed denial of service attacks. In Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0 (Vol. 3, pp. 2275-2280). IEEE.

[4] CERP Coordination Center, Cert Advisories: "CA-2000-01 denial-of-service developments:" http://www.cert.org/advisories/CA-2000-01.html; "CA-99-17 denial-of service tools," http://www.cert.org/advisories/CA-99-17-denial-of-servicetools.html; "CA-98-1 3-tcp-denial-of-service: vulnerability in certain TCP/IP implementations," http://www.cert.org/advisories/CA-98-13-tcp-denial-of-service.html.

[5] B. Martin, "Have script, will destroy (lessons in DOS)," Feb. 2000, http://www.attrition.org.

[6] CERP Coordination Center, "Results of the distributed systems intruder tools workshop," Nov. 1999, http://www.cert.org/reports/dsit-workshop.pdf

[7] B. McCarty, "Botnets: big and bigger," IEEE Security and Privacy, vol. 1, no. 4, pp. 87–90, 2003.

[8] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, "Know your Enemy: Tracking Botnets," http://www.honeynet.org/papers/bots.

[9] F. C. Freiling, T. Holz, and G. Wicherski, "Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks," in Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS '05), vol. 3679 of Lecture Notes in Computer Science, pp. 319–335, Springer, Milan, Italy, September 2005.

[10] K. Pappas, “Back to basics to fight botnets,” Communications News, vol. 45, no. 5, p. 12, 2008.

[11] P. Sroufe, S. Phithakkitnukoon, R. Dantu, and J. Cangussu, “Email shape analysis for spam botnet detection,” in Proceedings of the 6th IEEE Consumer Communications and Networking Conference (CCNC ’09), pp. 1–2, Las Vegas, Nev, USA, January 2009.

[12] K. Chiang and L. Lloyd, “A case study of the restock rootkit and spam bot,” in Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets, p. 10, Cambridge, Mass, USA, 2007.

[13] A. Brodsky and D. Brodsky, “A distributed content independent method for spam detection,” in Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets, p. 3, Cambridge, Mass, USA, 2007.

[14] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, “Spamming botnets: signatures and characteristics,” in Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM ’08), vol. 38, pp. 171–182, Seattle, Wash, USA, August 2008.

[15] Liu, J., Xiao, Y., Ghaboosi, K., Deng, H., & Zhang, J. (2009). Botnet: classification, attacks, detection, tracing, and preventive measures. EURASIP journal on wireless communications and networking, 2009, 1-11.

[16] Dave, K. T. (2013). Brute-force attack ‘seeking but distressing’. Int. J. Innov. Eng. Technol. Brute-force, 2(3), 75-78.

[17] Aamir, M., Rizvi, S. S. H., Hashmani, M. A., Zubair, M., & Ahmad, J. (2021). Machine learning classification of port scanning and DDoS attacks: A comparative analysis. Mehran University Research Journal Of Engineering & Technology, 40(1), 215-229.

[18] “NMAP: The Network Mapper – Free Security Scanner”, https://nmap.org/, [Last Visited on 27th November 2023.

[19] Beheshti, Z., & Shamsuddin, S. M. H. (2013). A review of population-based meta-heuristic algorithms. Int. j. adv. soft comput. appl, 5(1), 1-35.

[20] Yusof, I., & Pathan, A. S. K. (2014, November). Preventing persistent Cross-Site Scripting (XSS) attack by applying pattern filtering approach. In The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M) (pp. 1-6). IEEE.

[21] Gupta, M. K., Govil, M. C., & Singh, G. (2015, July). Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications. In 2015 12th international joint conference on computer science and software engineering (JCSSE) (pp. 162-167). IEEE.

[22] Van Gundy, M., & Chen, H. (2009, February). Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks. In NDSS.

[23] Khari, M., & Sangwan, P. (2016, March). Web-application attacks: A survey. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 2187-2191). IEEE.

[24] Nasteski, V. (2017). An overview of the supervised machine learning methods. Horizons. b, 4, 51-62.

[25] Talwar, A., & Kumar, Y. (2013). Machine Learning: An artificial intelligence methodology. International Journal of Engineering and Computer Science, 2(12), 3400-3404.

[26] Caruana, R., & Niculescu-Mizil, A. (2006, June). An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning (pp. 161-168).

[27] Sandhya, N., & Charanjeet, K. R. (2016). A review on machine learning techniques. International Journal on Recent and Innovation Trends in Computing and Communication, 4(3), 451-458.

[28] Taunk, K., De, S., Verma, S., & Swetapadma, A. (2019, May). A brief review of nearest neighbor algorithm for learning and classification. In 2019 international conference on intelligent computing and control systems (ICCS) (pp. 1255-1260). IEEE.

[29] Asselman, A., Khaldi, M., & Aammou, S. (2023). Enhancing the prediction of student performance based on the machine learning XGBoost algorithm. Interactive Learning Environments, 31(6), 3360-3379.

[30] Boswell, D. (2002). Introduction to support vector machines. Departement of Computer Science and Engineering University of California San Diego, 11.

[31] Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2016). Introduction to Anaconda and Python: Installation and setup. Quant. Methods Psychol, 16(5), S3-S11.

[32] Szumelda, P., Orzechowski, N., Rawski, M., & Janicki, A. (2022, June). Vhs-22–a very heterogeneous set of network traffic data for threat detection. In Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference (pp. 72-78).

[33] Scikit-learn, S. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection. GridSearchCV.html (accessed on 28 November 2023).

[34] Scikit-learn, S. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection. RandomizedSearchCV.html (accessed on 28 November 2023).

[35] Scikit-optimize, S. Available online: https://scikit-optimize.github.io/stable/modules /generated/skopt. *BayesSearchCV.html (accessed on 28 November 2023).*

[36] Kodinariya, Trupti M., and Prashant R. Makwana. "Review on determining number of Cluster in K-Means Clustering." *International Journal* 1.6 (2013): 90-95.

CURRICULUM VITAE

2001 – 2007	B.Sc., Computer Education And Instructional Technology, Middle East Technical University, Ankara, TURKEY
2014 – Present	M.Sc., Electrical and Computer Engineering, Abdullah Gül University, Kayseri, TURKEY