Real-Time Detection and Mitigation of DDoS Attacks using Machine Learning Approach

Gutema Bunturi Midakso

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of

Master of Science in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September, 2023

Adama, Ethiopia

Real-Time Detection and Mitigation of DDoS Attacks using Machine Learning Approach

Gutema Bunturi Midakso

Advisor(s) Biruk Yirga (PhD)

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master of Science in computer science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September, 2023

Adama, Ethiopia

Declaration

I hereby declare that this Master Thesis entitled "**Real-Time Detection and Mitigation of DDoS Attacks using Machine Learning Approach"** is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

_____	_____	_____
Name of student	Signature	Date

Recommendation of Advisors

I/we, the advisor(s) of this thesis, hereby certify that I/we have read the revised version of the thesis entitled "**Real-Time Detection and Mitigation of DDoS Attacks using Machine Learning Approach**" prepared under my/our guidance by **Gutema Bunturi Midakso** submitted in partial fulfillment of the requirements for the degree of Masters of Science in computer science and Engineering. Therefore, I/we recommend the submission of the revised version of the thesis to the department following the applicable procedures.

_____ _____ _____

Advisor Signature Date

Approval Page

I/we, the advisors of the thesis entitled "**Real-Time Detection and Mitigation of DDoS Attacks using Machine Learning Approach**" developed by **Gutema Bunturi Midakso** hereby certify that the recommendations and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Advisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by **Gutema Bunturi Midakso** have read and evaluated the thesis entitled "**Real-Time Detection and Mitigation of DDoS Attacks using Machine Learning Approach**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

Chairperson	Signature	Date
Internal Examiner	Signature	Date
Dr. Hussien Seid		Oct. 09, 2023
External Examiner	Signature	Date

Final approval and acceptance of the thesis are contingent upon the submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGMENTS

I would like to express my deepest gratitude and acknowledgment to God for granting me strength, perseverance, and guidance throughout this journey of academic pursuit. I am truly thankful for the blessings and opportunities that have shaped this work.

I extend my appreciation to Adama Science and Technology University for providing the facility for me to undertake this research. The conducive academic environment and resources played a pivotal role in conducting this study.

I am indebted to my advisor, **Dr. Biruk Yirga**, for his invaluable mentorship, unwavering support, and insightful guidance. His expertise and dedication have been instrumental in shaping the direction and quality of this thesis.

Lastly, I am profoundly grateful to my family for their unconditional love, encouragement, and unwavering belief in me. Their steadfast support has been the cornerstone of my academic journey.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACC: Accuracy

DDoS: Distributed Denial of Service

DoS: Denial of Service

DT: Decision Tree

FN: False Negative

FP: False Positive

HTTP: Hyper-text transfer protocol

ICMP: Internet control message protocol

IEEE: Institute of Electrical and Electronics Engineers

IoT: Internet of Things

IP: Internet protocol

KNN: K-Nearest Neighbor

LAN: Local area networks

LDoS: Low-Rate Denial-of-Service

LR: Logistic regression

MAC: Media access control

MAN: Metropolitan area network

ML: Machine Learning

NB: Naïve Bayes

OMNET++: Objective Modular Network Testbed in C++

RF: Random forest

SDS: Smart Signature Dataset

SMOTE: Synthetic Minority Over-sampling Technique

SVM: Support Vector Machine

TCP/IP: Transmission Control Protocol/Internet Protocol

TN: True negative

TP: True positive

UDP: User Datagram Protocol

WAN: Wide-area network

Wi-Fi: Wireless Fidelity

Abstract

Among the diverse threats that wait in the digital realm, Distributed Denial of Service (DDoS) attacks emerge large as an insidious and ever-evolving menace. A Distributed Denial of Service attack is a malicious attempt to affect the availability of a targeted system, such as a website or application, to legitimate end users. The motivation behind this thesis is to harness the potential of machine learning to develop a system capable of real-time detection and mitigation of DDoS attacks. By doing so, we aim to fortify the cybersecurity landscape and ensure that organizations and individuals can navigate the digital world with greater resilience and security in the face of this persistent threat. The core issues the research aim to tackle is the imperative for an advanced and real-time DDoS detection and mitigation system, driven by Machine Learning (ML) methodologies. Even though there are various other security issue that must be addressed, safeguard network against Distributed Denial-of-Service (DDoS) attacks, ensuring that legitimate users can access the service while mitigating malicious packets from infiltrating the network is the primary scope of the research. A proposed real time detection and mitigation model for DDoS attack predict the class of traffic and mitigate it with IP blocking mechanism. The dataset for this study was obtained from real world public data source. The study proposed 6 models: SVM, DT, RF, NB, KNN and LR and Bidirectional feature selection techniques. The proposed model was evaluated using performance evaluation metrics and SVM outperformed other models on our dataset achieving accuracy of 97.3%. Finally, the model with better performance result are selected as the model with excellent detection result of DDoS attack.

Key words: Distributed Denial of Service, Machine Learning, Support Vector Machine*, Decision Tree, Random forest, Naïve Bayes, K-Nearest Neighbor, Logistic regression*

CHAPTER ONE

1. INTRODUCTION

1.1 Background of the Study

In our current digital landscape, marked by global connectivity and an escalating reliance on online services, the preservation of network security has emerged as a paramount concern. Among the diverse threats that wait in the digital realm, Distributed Denial of Service (DDoS) attacks emerge large as an insidious and ever-evolving menace. A Distributed Denial of Service (DDoS) attack is a malicious effort to disrupt the accessibility of a specific system, like a website or application, for genuine users. In these instances, attackers usually create substantial amounts of data packets or requests with the ultimate goal of overwhelming the intended system.

DDoS attacks are executed using networks encompassing Internet-connected machines. These networks are composed of computers and various devices, including IoT devices, that have been infected with malicious software, allowing them to be manipulated from a remote location by an attacker. These individual devices are commonly known as bots or zombies, and a collective group of these bots is referred to as a botnet. Once a botnet has been established, the attacker gains the capability to carry out an attack by sending remote instructions to each bot.

When a victim's server or network is targeted by the botnet, each bot messages requests to the target's IP address, potentially causing the server or network to become overwhelmed, ultimately resulting in a denial-of-service for regular traffic.

These carefully planned attacks, usually carried out by bad actors, flood the networks or services they target with an enormous amount of traffic, causing them to stop working. The results are serious and include significant money losses, damage to the reputation of a brand, and the interruption of important services.

Historically, DDoS attacks were countered by observing network traffic and making decisions using statistical analysis, like anomaly detection, or specific rules for each packet. These techniques have been reasonably effective in preventing attacks with well-known or expected behaviors.

Nevertheless, all the decision criteria and thresholds had to be set manually, relying on expert knowledge or patterns from past attacks. Lately, this approach has encountered difficulties due to the emergence of less predictable attack methods that can evade these established rules.

At its essence, this research encompasses a comprehensive set of goals aimed at leveraging machine learning to enhance network security. A primary objective is to develop machine learning models specifically tuned to precisely distinguish between the normal fluctuations in network traffic and the sudden spikes associated with DDoS attacks. These models, acting like digital sentinels, continually adapt by learning from various datasets while defense mechanism against evolving attack should be put in place. (K. Kumari & Mrunalini, 2022).

In addition to the careful detection process, this research also explores ways to fight against DDoS attacks. This strategy is like a part of the machine learning model, and it start working as soon as an attack is detected. Their job is to use information from past attacks and what's happening right now to make smart decisions. This flexible response not only stops the immediate danger but also makes the system better at handling future attacks by learning and changing over time.

The core challenge in this research centers around carefully detecting and mitigating the network traffic spikes and down. To do this, we use publicly available real-world data and carefully train our machine learning model after the preliminary data processing has been done. The next step will lay down to the mitigation stage where, network traffic having the pattern attack is denied the access to the network and their respective IP address are stored. It's crucial to make sure that the solution doesn't just look good on paper but can actually withstand and outperform the constantly changing tactics used by DDoS attackers.

Moreover, the thesis deeply considers the practicality of the proposed solution. While theoretical excellence is worthy, it is paramount that the solution can be seamlessly integrated into the complex wall-hanging of real-world network environments.

By following these multifaceted objectives, this thesis aspires to carve a significant function in the cybersecurity landscape. It endeavors to helper in an era where machine learning is not just an auxiliary tool but a central pillar in the unending battle against DDoS threats. Through its difficult research, this work seeks to furnish invaluable insights and tangible, practical solutions that empower organizations and service providers to safeguard their digital infrastructures. In doing

so, it reinforces the resilience of online services, ensuring the uninterrupted availability and reliability of the digital backbone that supports our interconnected world. Ultimately, the research aspires to be a cornerstone in the ongoing endeavor to fortify our digital future against the relentless and ever-adapting adversaries arranging DDoS attacks(Najafimehr et al., 2023).

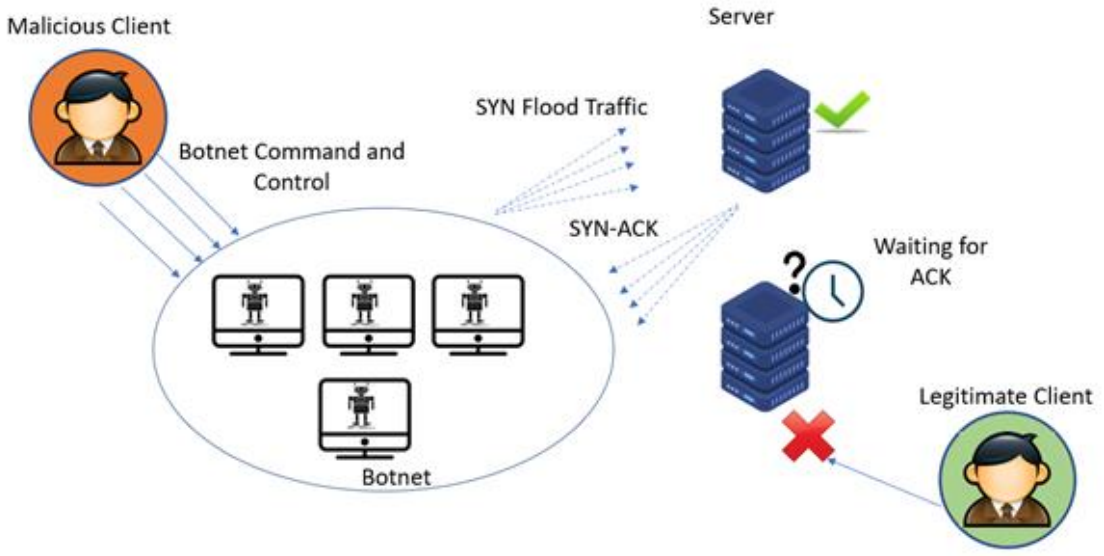

Figure 1: Graphical depiction of how DDoS attacks works

1.2 Motivational of the Study

In today's digital age, the threat of Distributed Denial of Service (DDoS) attacks emerges larger than ever, posing a challenge to organizations and individuals alike. These attacks, characterized by their capacity to overwhelm network resources, not only disrupt services but also impose severe financial and reputational damage. With our increasing reliance on digital infrastructure, the urgency of devising robust DDoS detection and mitigation mechanisms cannot be excessive.

DDoS attacks are not only on the rise but are also becoming more intricate, making it crucial to explore innovative strategies to combat this evolving menace. Traditional defenses, reliant on signature-based solutions and network rules, have become inadequate in the face of rapidly changing attack tactics. It is within this context that Machine Learning (ML) emerges as a game-changing technology in the realm of cybersecurity. ML's inherent capacity to analyze extensive datasets and discern patterns in real-time offers a promising road for DDoS detection and

mitigation. By adapting to and learning from the evolving strategies of attackers, ML models provide a dynamic and proactive defense mechanism.

By saying so, the primary motivation behind conducting this research is to capture the dynamic nature of DDoS attack and mitigate it so that the legitimate users can access the service they want acquire anytime from any place without any worrisome. Furthermore, the research is conducted so that companies and different organization could be protected from unnecessary losses and expenses. Other reason behind conducting this research is to develop the experience of conducting viable research to tackle the problems arising in the communities we are living.

Yet, the primary challenge lies in achieving real-time detection and mitigation, DDoS attacks can be initiated within seconds, allowing minimal time for human intervention. Hence, the core motivation of this thesis is to harness the potential of machine learning to develop a system capable of real-time detection and mitigation of DDoS attacks. By doing so, we aim to fortify the cybersecurity landscape and ensure that organizations and individuals can navigate the digital world with greater resilience and security in the face of this persistent threat.

1.3 Statement of the Problem

In the digital era, the danger of Distributed Denial of Service (DDoS) attacks has reached alarming proportions. These attacks, which involve overwhelming online systems with a flood of malicious traffic, have the potential to disrupt vital online services and impose substantial economic and reputational damage. Traditional defense mechanisms, characterized by static rules and signatures, are becoming increasingly ineffective in countering these ever-evolving attacks.

The core issue we aim to tackle in this thesis is the hunger for an advanced and real-time DDoS detection and mitigation system, driven by Machine Learning (ML) methodologies. This system must possess the capability to quickly identify and neutralize DDoS attacks in progress, thereby ensuring the uninterrupted availability and integrity of online resources in a dynamic threat environment.

At the heart of this challenge lies the development of an ML-powered solution that not only adapts to emerging attack tactics but also responds with minimal potential. This involves the creation of intelligent algorithms that can continuously learn and evolve, providing a robust defense against

the persistent threat of DDoS attacks. In essence, our mission is to fortify digital infrastructure against these threats, making the online world safer and more resilient for organizations and individuals alike.

1.4 Research Questions

To reduce and prevent this denial of service by attackers, detecting and mitigating the nature of attacker has paramount advantage. The following research questions are the main research questions that should be answered by this research work:

- What are the key features that are determinant in the detection of DDoS attack in Machine Learning models?
- Which classification algorithms will best detect DDoS attacks?

1.5 General and Specific Objectives

1.5.1 General Objective

The primary aim of this research is to create a model that employs a machine learning approach for the detection and mitigation of DDoS attack.

1.5.2 Specific Objectives

- Assessing the currently employed strategies, policies, and algorithms aimed at ensuring service availability of service.
- Collecting dataset form real world public data sources.
- Applying data preprocessing techniques to clear up and prepare our dataset for model training
- Applying feature selection techniques to determine important features for our model and ignore the insignificant one.
- Training the machine learning models with preprocessed dataset.
- Evaluate the performance of the model using machine learning evaluation metrics

1.6 Significance of the Study

This study holds significant importance in the field of cybersecurity for several compelling reasons. Firstly, it addresses a critical security concern by focusing on the detection and mitigation

of Distributed Denial of Service (DDoS) attacks. DDoS attacks are a pervasive cyber threat with the potential for devastating consequences. They can disrupt communication channels across various networks, jeopardizing essential functions and services. By developing effective methods to identify and counteract these attacks, this research aims to enhance the overall security and resilience of digital networks.

Moreover, this study explores the application of machine learning techniques, which have shown great promise in the realm of intrusion detection. Machine learning can analyze network traffic patterns and identify abnormal behaviors indicative of DDoS attacks. By leveraging these techniques, the research seeks to provide a proactive and real-time defense against such attacks, ultimately contributing to the availability and reliability of digital networks in general.

Furthermore, the study acknowledges the broader implications of its findings. Effective communication and the security of digital networks are essential for various applications, including data transmission, online services, and critical infrastructure operations. By enhancing the security of these networks through DDoS attack detection and mitigation, this research indirectly supports these vital functions, fostering a more secure and resilient digital system.

In summary, this study not only addresses a pressing security concern but also aligns with broader goals of improving the reliability and security of digital networks. By developing and implementing real-time security mechanisms against DDoS attack. It endeavors to support the foundation upon which the future of digital infrastructure, communication systems, and online services relies.

1.7 Delimitation Scope

The primary objective of this research paper is to safeguard network against Distributed Denial-of-Service (DDoS) attacks, ensuring that legitimate users can access the service while mitigating malicious packets from infiltrating the network. Network may face various security threats, including Broadcast Tampering, Malware, Spamming, Black Hole Attack, Worm Hole Attack, Greedy Drivers, Sybil Attack, and more. However, our study concentrates solely on addressing the issue of Slowloris and Volumetric DDoS attacks.

1.8 Thesis Organization

This thesis is structured into five comprehensive chapters, each dedicated to addressing various aspects of the research. In Chapter One, the study begins with a detailed introduction encompassing the problem statement, motivation for the research, specified objectives, research methodology, significance of the study, and the scope of the work.

Chapter Two delves into a concise yet comprehensive review of relevant literature and related work in the field. It provides readers with an overview of the research area, highlighting key concepts, prior research efforts, and gaps in previous studies that this research aims to address.

Chapter Three outlines the methodology employed in the study, including DDoS attack classification modeling, the evaluation of classification models, and the utilization of performance evaluation metrics. This chapter also covers the overall working principles and architectural details of the thesis.

Chapter Four presents the proposed solution, including the architecture of the DDoS attack detection and mitigation model, data understanding, preprocessing implementations, model testing and evaluation, machine learning model implementations, and hyperparameter tuning.

Moving on to Chapter Five, this section focuses on results evaluation and discussions, encompassing aspects like dataset class distributions, model performance and evaluation results, and the outcomes of hyperparameter tuning. This chapter provides a comprehensive analysis of the research findings.

Finally, the concluding section of the thesis, Chapter Six, summarizes the key takeaways of the research, offers recommendations, and outlines potential directions for future research in this field. This organized structure ensures a clear presentation of the research's progression from introduction to outcomes, facilitating a detailed understanding for readers.

CHAPTER TWO

2. LITERATURE AND RELATED WORK

This chapter is organized into two main sections. The first section serves as an introduction, providing a brief overview of the subject area explored in this thesis. It primarily focuses on introducing the concepts of network security, DDoS (Distributed Denial of Service) attacks, and the significance of real-time detection and mitigation. Since the study seeks to develop strategies applicable to a wide range of network environments, this introductory section lays the groundwork by highlighting the fundamental challenges posed by DDoS attacks in modern digital networks.

The second section delves into a detailed examination of the current landscape of DDoS attacks and the defensive strategies employed against DDoS attack in network security. It encompasses a comprehensive review of academic research and materials from security organizations, focusing on methods for detecting and countering DDoS attacks. Before exploring into the study's specific objectives, it's essential to establish a foundational understanding of network security architecture, which serves as the backdrop for addressing DDoS attacks through Machine Learning-based techniques in subsequent chapters.

2.1 Overview of Traditional Network

In the ever-evolving landscape of modern networking, the study of real-time detection and mitigation of Distributed Denial of Service (DDoS) attacks assumes a critical role. This thesis explores into the complex domain of network security, where the fusion of Machine Learning and DDoS mitigation takes center stage. This research transcends the confines of specific network types and explores the broader realm of network configurations and their unique characteristics.

Networks, characterized by their diverse and dynamic nature, serve as the backbone of modern communication. They facilitate the seamless exchange of data and information across a wide array of devices and applications. However, this interconnectedness also brings forth security challenges, particularly in the face of DDoS attacks, which disrupt the availability of services and pose significant threats.

In response to these evolving security concerns, this research endeavors to develop real-time detection and mitigation strategies using Machine Learning techniques. By leveraging the power

of data-driven algorithms, the aim is to boost network security, ensuring the uninterrupted flow of information in an increasingly interconnected digital world.

Generally, this traditional network overview sets the stage for a comprehensive exploration of DDoS attack detection and mitigation, underscoring its relevance and significance in the broader landscape of network security enhancement.

2.1.1 Architecture of Network

The Structural Communication Network Architecture is a pivotal framework that underpins the intricate web of modern communication systems. It represents the fundamental blueprint upon which our interconnected digital world is built. At its core, this architecture encompasses the intricate hierarchy of network components, defining how data and information flow seamlessly across various levels and layers. It provides a structured framework for the organization, transmission, and reception of data, facilitating the efficient exchange of information among a wide array of devices and applications.

At the heart of this architecture are the foundational building blocks, which include routers, switches, gateways, and protocols. These components work in agreement to ensure that data packets are routed to their intended destinations with precision and speed. The hierarchical structure of the network, often ensures that data is efficiently managed, controlled, and distributed. This approach optimizes the performance and scalability of communication systems, allowing for the seamless expansion of networks to accommodate growing demands.

Moreover, the Structural Communication Network Architecture is inherently adaptable, evolving alongside technological advancements. It integrates emerging technologies like cloud computing, edge computing, and the Internet of Things (IoT) into its framework, enabling the network to adapt to the dynamic requirements of the digital age. As a result, this architecture not only forms the backbone of our everyday communication but also serves as the enabler for innovative solutions that drive progress in various domains, from telecommunication and finance to healthcare and transportation(*IEEE Instrumentation & Measurement Magazine Network Measurement: The Context,* 2008).

Generally, the Structural Communication Network Architecture is the base of modern connectivity, arranging the work of data and information exchange in our interconnected world. Its adaptability and scalability ensure that it remains at the forefront of technological evolution, supporting the growth and innovation that define our digital era.

2.2. Challenges in Network Communication

2.2.1 Communication in Network

Networks and Communication refer to the interlinked systems and devices that enable the transfer of data and information between different endpoints. Essential components of network and communication systems include hardware elements such as computers, routers, switches, modems, and software protocols that govern the data exchange between these devices.

Networks can be classified into different types, such as local area networks (LANs), wide area networks (WANs), metropolitan area networks (MANs), and wireless networks (such as Wi-Fi and cellular networks). Communication protocols like TCP/IP and HTTP facilitate device communication by establishing a shared language and a set of guidelines for data interchange.

Technological progress has led to the emergence of intricate communication networks like the Internet, fundamentally altering the way people communicate and obtain information. The extensive utilization of these networks has enabled enhanced cooperation, increased productivity, and broader availability of information and resources(Jonsson et al., n.d.).

2.2.2 Technical Challenges in Computer Network

These challenges pertain to the technical issues that need to be addressed. Here is a list of some of the technical challenges:

a) Network Management: Due to the rapid and constant changes in the network's structure and conditions, using configurations like trees is unreasonable, as these structures cannot be established and maintained as quickly as the topology evolves.

b) Congestion and Collision Control: The challenge is further compounded by an expansive network size with no limitations. In rural areas and during nighttime in urban settings, network capacity tends to be minimal. As a result, network congestion is a common occurrence, particularly during peak usage hours when network loads are exceptionally heavy, leading to network overcrowding and collisions.

c) Environmental Impact: Networks utilize electromagnetic waves for communication, and these waves are influenced by the surrounding environment. Hence, when deploying such networks, it is essential to consider the environmental factors that can affect them(Bazzan, n.d.).

2.2.3 Security Challenges

In the context of networks, it is essential to guarantee the security of information exchanged between nodes, ensuring that it remains untouched and unaltered by potential attackers. Moreover, the responsibility of each user is crucial for timely and accurate communication within the network. The absence of centralized management exposes such networks to various security risks. To simplify identification and categorization, these threats are classified into five distinct categories, each representing a different attack stage and priority level within its respective class.

2.2.4 Communication Challenges

While research efforts have generally focused on the security of networks against DDoS attack, there is still room for improvement in developing solutions that effectively safeguard these networks from adversaries and malicious actors. The goal is to attain a satisfactory level of security, ensuring both sender and receiver of data are uninterrupted.

A. Security

Managing trust and ensuring the security of message content poses a challenge in network communication. It's essential to rapidly verify the received information's content to make it promptly usable.

B. Authentication

The authentication service focuses on ensuring the authenticity of communication entities, allowing nodes to respond to events by broadcasting messages solely from legitimate senders. Hence, it becomes necessary to verify the authenticity of these message senders(Farik et al., 2016).

C. Integrity

The integrity service addresses the correctness of a series of messages, ensuring that messages are received in the same state as they were sent, with no alterations, additions, changes in sequence, or repeated transmissions.

D. Confidentiality

This service ensures the secrecy of communication content, safeguarding node's privacy from unauthorized onlookers. Confidentiality is closely tied to security, and since network are valuable assets, users who own them must take steps to safeguard their personal data. Various methods are available to protect this information, such as accumulating data over an extended period from multiple source nodes and analyzing it(Bourke & Wessely, 2008).

E. Accessibility

Various forms of attacks can lead to a loss or reduction in accessibility. Even a robust communication system may still experience certain attacks, like denial of service, which can disrupt the network. Hence, availability should be reinforced through additional measures, with an essential aspect of network security being the use of digital signatures as a foundational component. Whether it's infrastructure communications or inter-network communication, authentication via signatures is a fundamental security necessity(*An Overview of Computer Security*, n.d.).

F. Scalability

Scalability, in this context, signifies the capacity for an increase in the number of users and/or traffic volume with minimal performance decline or network interruptions, all without the need to alter the fundamental system elements and protocols. It represents a network's capability to seamlessly accommodate a growing number of communicating traffic, avoiding disruptions or data transfer losses, even though it may lead to increased administrative complexity and reduced network efficiency(Bondi, 2000).

G. Reliability

Due to the constrained time frame for communication, guaranteeing dependable receipt and acknowledgment of messages between entities poses a formidable challenge. In normal networks, most of the messages sent are periodic broadcast messages intended to update nearby entities about a sender's status. Consequently, for broadcast messages, a heightened level of responsibility becomes imperative.

H. Media Access Control

In order to establish extensive networks, modifications are necessary at the media access control (MAC) layer. The primary role of the MAC layer is to manage access to the shared wireless channel. Without a system in place to synchronize data transmissions, a significant number of collisions would happen, leading to the loss of data.(Christopher U et al., 2019).

2.3 Overview of DDoS Attack

A Distributed Denial of Service (DDoS) attack is a malicious effort to disrupt the normal operation of a network, service, or website by flooding it with an excessive amount of internet traffic. These attacks employ numerous compromised computers or devices, which collectively form a botnet. This botnet generates an immense volume of traffic that inundates the target's bandwidth, processing capacity, or other essential resources.(Gu & Liu, n.d.).

These attacks come in various forms, including volumetric attacks that flood the target, TCP state exhaustion attacks that exhaust connection state tables, and application layer attacks that target vulnerabilities in specific applications. DDoS attacks can be short-lived or prolonged, and they can have diverse motivations, ranging from financial gain to conceptual reasons. A successful DDoS attack can have serious consequences, leading to financial setbacks, harm to a company's reputation, and disruptions in its operations. Organizations use various defense techniques, including traffic rate restriction, IP blocking and load balancing, to defend against DDoS attacks. These attacks are illegal in most jurisdictions, and those responsible for launching them can face legal consequences. As DDoS attack techniques continually evolve, constant care and adaptation are essential for effective defense(Farnham et al., n.d.).

2.3.1 Categories of DDoS Attack

Distributed Denial of Service (DDoS) attacks encompass a range of major types, each with distinct tactics to disrupt the communication system. DDoS attacks can typically be grouped into three primary attack categories. (Majeed Alhammadi et al., 2022):

2.3.1.1 Volumetric DDoS Attacks

Volumetric DDoS attacks aim to overpower the resource's capabilities. Servers receive an excessive number of requests, networks become swamped with traffic, and databases may be flooded with calls.

When it comes to the internet, a DDoS attack aims to saturate the targeted site's available bandwidth, with the attack's intensity often measured in bits per second. Volumetric DDoS attacks encompass various flood attacks, such as UDP floods and TCP floods, as well as the misuse of applications.

2.3.1.2 Protocol DDoS Attacks

Rather than relying solely on massive traffic volume, protocol-based DDoS attacks exploit protocols to inundate a particular resource, typically a server, but occasionally firewalls or load balancers. These attacks are commonly assessed by their rate in packets per second. Some familiar examples of protocol-based DDoS attacks include the IP Null attack and slowloris.

2.3.1.3 Application DDoS Attacks

Application DDoS attacks zero in on weaknesses within applications, aiming to induce a failure within the application itself. In contrast to attacks that primarily disrupt infrastructure, this type of attack hones in on Layer 7 software. Nevertheless, it can lead to overwhelmed CPUs or depleted memory, impacting both the server and other applications. The scale of an application DDoS attack is assessed by the rate of requests per second, with HTTP Flood attacks being a notable example within this category.

2.3.2 DDoS Mitigation Techniques

Mitigation techniques against Distributed Denial of Service (DDoS) attacks are crucial for safeguarding online services and networks. These techniques are designed to identify and counteract malicious traffic, ensuring that legitimate users can access resources without interruption. The following are approaches to mitigate DDoS attack(Mahajan & Sachdeva, 2013).

2.3.2.1 Load Balancing

The primary purpose of load balancing is to evenly spread incoming network traffic among multiple servers or resources, preventing any single server from becoming overloaded. It can help in mitigating DDoS attacks to some extent by distributing the attack traffic among multiple servers. Incoming traffic is routed through a load balancer and the load balancer evenly distributes the traffic to multiple backend servers. This distribution helps prevent any single server from becoming a bottleneck, making it harder for attackers to overwhelm a specific resource.

However, load balancing alone is not sufficient to mitigate large-scale DDoS attacks because it cannot differentiate between legitimate and malicious traffic. It merely spreads traffic evenly.

2.3.2.2 Rate Limiting

Rate limiting is a DDoS mitigation technique that involves controlling the rate at which traffic is allowed to enter a network or server. It helps to protect against certain types of DDoS attacks that rely on overwhelming resources with high traffic volumes. If the incoming traffic exceeds a predefined threshold, traffic that exceeds the limit is either dropped or slowed down.

This prevents the server from being inundated with excessive traffic, protecting it from becoming overwhelmed. Rate limiting is effective against some types of DDoS attacks but may not be sufficient against highly distributed and varied attacks(Mahjabin et al., 2017).

2.3.2.3 IP Blocking

IP blocking is a more direct and targeted DDoS mitigation technique that involves identifying malicious IP addresses and blocking them from accessing the protected resource. Traffic patterns are analyzed, and suspicious or malicious IP addresses are identified. The identified IPs are then blocked at the network or server level, preventing them from sending further traffic to the resource. This helps to stop the attack traffic from specific sources, reducing the impact of the DDoS attack.

IP blocking can be effective against known attackers or attackers coming from specific IP ranges. However, it can also have limitations, such as being less effective against attacks that use a large number of compromised machines with constantly changing IP addresses.

2.4 Stream Processing

Stream processing is a data processing technique that involves the real-time analysis of continuously flowing data streams. It is used to handle and analyze data as it is generated, making it a critical component for achieving real-time capabilities

Real-time DDoS detection and mitigation require the use of stream processing techniques. Employing stream processing frameworks like Apache Kafka, Apache Flink, or Apache Storm to continuously analyze network traffic data as it flows in real-time.

2.4.1 How Does Stream Processing Function/Work?

Data from different source are all examples of data that can benefit from Stream Processing. Publisher/subscriber (also known as pub/sub) and source/sink are two prevalent paradigms. A publisher or source generates data and events, which are provided to a Stream Processing Application, where they are enhanced, tested against fraud detection algorithms, or otherwise altered before being sent to a subscriber or sink. Apache Kafka, Apache Storm, TCP connections, and in-memory Data Grids are some of the most frequent sources and sink on the technical side.

2.5 Overview of Machine Learning

Numerous advancements have emerged as a result of technological progress in areas such as the internet, computing, and related fields. These developments have impacted sectors including security, law, medicine, agriculture, and e-commerce, all with the aim of satisfying customers, saving time and resources, and increasing profitability. Presently, many businesses rely on computer systems and artificial intelligence for their operations.

Within the realm of computer science and artificial intelligence, a branch known as "machine learning" has gained prominence. Machine learning leverages vast sets of organizational data to tackle complex challenges, especially when dealing with extensive data, and plays a pivotal role in identifying specific patterns within this data(Aldoseri et al., 2023).

Machine learning involves the use of copious amounts of data and algorithms to emulate how individuals learn from experience and enhance their performance. Instead of explicit programming, this approach entails teaching machines what to do and enabling them to carry out tasks automatically based on acquired experience.

Machine learning can be categorized into four primary approaches: supervised, unsupervised, semi-supervised, and reinforcement learning methods(Oladipupo, 2010).

2.5.1 Supervised Machine Learning

Supervised learning stands as a category within machine learning, distinguished by its approach of constructing a machine-learning model with the guidance of known labels. This algorithm is characterized by the presence of a target or dependent attribute that is defined by a range of classifiers or independent attributes.

The process of supervised learning relies on data that has already been labeled or categorized(Sarker, 2021).

Subsequently, the algorithm employs this labeled data to create rules and then applies these newly acquired rules to categorize unobserved data whose outputs are unknown. In supervised learning, the primary difficulties often revolve around classification and regression tasks. Prominent instances of supervised learning techniques include Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, K-Nearest Neighbor, and Logistic Regression.

2.5.2 Unsupervised Learning

The process of handling unlabeled data is referred to as unsupervised learning. This particular algorithm is exceptionally effective for analyzing data, detecting patterns and trends, even when the category of the data point is unknown. It's a method of learning that can be employed to cluster or categorize a given dataset into distinct groups. An example of unsupervised learning frequently used is the K-means algorithm(Karhunen et al., n.d.).

2.5.3 Semi-Supervised machine learning

Semi-supervised machine learning represents a specific approach within the realm of machine learning that leverages a combination of labeled and unlabeled data during the training process. This method lies between supervised and unsupervised learning, employing a limited set of labeled data to initiate model training while utilizing a substantial volume of unlabeled data to enhance its precision. Semi-supervised learning algorithms find practical applications in areas like natural language processing, image analysis, and speech recognition(C A Padmanabha Reddy et al., 2018).

2.5.4 Reinforcement Learning

Reinforcement learning is a form of machine learning that empowers an entity to acquire knowledge through interacting with its surroundings, making decisions, and gaining rewards in response to those decisions. It operates on the principle of learning from experimentation, where the entity gains insights from its errors and steadily enhances its abilities over time. The ultimate aim of reinforcement learning is to optimize the total reward obtained by the entity throughout its learning process(Srinivas & Swapna, n.d.).

2.5.5 Some Popular Classification Algorithms

This section presents some of the common ML classification algorithms with their strengths and weaknesses(Vijaya et al., n.d.).

2.5.5.1 Logistic Regression (LR): Logistic regression is a linear modeling method employed to examine the relationship between a target variable and other variables by estimating probabilities through a logistic function. Initially, it was created by statisticians to explain population growth patterns in ecology. In logistic regression, input values (represented as χ) are linearly combined using 17 weights or coefficients to make predictions about an output value(y)(Daniel & Martin, 2023).

The output produced by logistic regression falls within the range of 0 to 1. Results closer to 1 indicate a stronger likelihood that the input variable belongs to a specific category, whereas values closer to 0 suggest that the input variable is less likely to belong to that category. The method's advantage lies in its straightforward mathematical interpretation. In the model, Y represents the classified output, B0 stands for the bias or intercept term, and B1 represents the coefficient for the single input value (χ).

Using a logistic regression model for classification is as simple as inputting values into the logistic regression equation and computing the result(*Chapter 4 Logistic Regression as a Classifier*, n.d.). However, one limitation of this algorithm is that it doesn't provide an intuitive understanding of classification as a linear combination of the input variables.

2.5.5.2 Random Forest: Random Forest is an ensemble classifier that employs a multitude of Decision Trees (DTs) on a dataset and combines their classifications. Collaborative methods encompass techniques that combine multiple machine learning models to produce more potent ones. The Random Forest method operates by utilizing a voting mechanism, where it predicts the output class by counting the votes from individual Decision Trees (DTs). The RF algorithm commences by randomly picking a portion of the training data and constructing a single DT from that subset.(Farnaaz & Jabbar, 2016).

This random sub-sample selection process is repeated multiple times, resulting in multiple DTs that collectively form a "forest." Once the forest is constructed, each individual tree makes its own

classification decision, and the final predicted class for an observation is determined by majority vote.

One of the primary advantages of Random Forest is its ability to mitigate the risk of overfitting. RF is easy to build and train, often yielding excellent results with a lower likelihood of overfitting compared to standalone Decision Trees.

The nature of Random Forest alleviates overfitting issues, eliminating the need for tree pruning. Additionally, its high classification accuracy, efficiency, and interpretability make it a preferred choice for various types of datasets.

2.5.5.3 Decision Tree: A decision tree is a classification model that employs a structure resembling a flowchart to make decisions based on input data. It segments the data into branches and connects results to leaf nodes. Decision trees are adaptable and serve purposes in both classification and regression tasks, providing easily understandable models(Bajaj et al., n.d.).

In decision support systems, decision trees are hierarchical models that illustrate decisions and their potential outcomes, considering factors such as random events, resource usage, and utility. This algorithmic approach depends on conditional control statements and falls within the realms of non-parametric and supervised learning, making it suitable for addressing both classification and regression tasks. The tree structure consists of a root node, branches, internal nodes, and leaf nodes, forming a hierarchical representation akin to a tree.

Decision trees have diverse applications across various domains. They are employed for solving classification and regression problems, following a tree-like structure akin to a flowchart, where predictions evolve through a series of feature-based splits, originating from a root node and culminating in decisions made at the leaves.

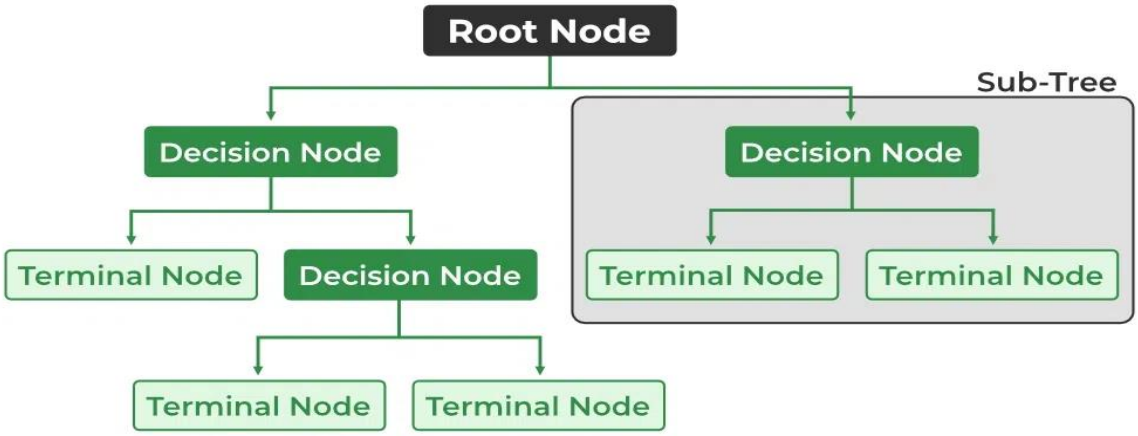

Figure 2: Decision tree structure

Advantages of the Decision Tree:

- ✓ Its simplicity makes it easily comprehensible, as it mirrors the decision-making process humans naturally follow in real-life scenarios.
- ✓ It proves highly valuable for addressing decision-related challenges.
- ✓ It encourages consideration of all potential outcomes for a given problem.
- ✓ It demands less data cleaning compared to alternative algorithms.

2.5.5.3 K-Nearest Neighbors: The K-Nearest Neighbor (KNN) technique is a widely utilized machine learning approach applied to tasks encompassing both classification and regression. It operates based on the premise that data points with similarities tend to share corresponding labels or values.

In the training phase, the KNN algorithm retains the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all the training instances, utilizing a selected distance metric such as the Euclidean distance.

Subsequently, the algorithm identifies the K nearest neighbors to the input data point based on their distances. In classification scenarios, it assigns the most frequently occurring class label among these K neighbors as the predicted label for the input data point. In regression, it calculates

either the average or weighted average of the target values associated with the K neighbors to predict the value for the input data point(Bajaj et al., n.d.).

Here are some of the advantages of using the k-nearest neighbors' algorithm:

- It's easy to understand and simple to implement
- It can be used for both classification and regression problems
- It's ideal for non-linear data since there's no assumption about underlying data
- It can naturally handle multi-class cases
- It can perform well with enough representative data

2.5.5.4 Support Vector Machine (SVM): A Support Vector Machine (SVM) is a supervised learning algorithm employed in machine learning to tackle classification and regression problems. SVMs excel in tackling binary and multiclass classification problems, which involve categorizing elements within a dataset into two distinct groups.

The main goal of a support vector machine algorithm is to determine the best line or decision boundary that segregates data points of distinct classes. In high-dimensional feature spaces, this boundary is known as a hyperplane. The central concept is to maximize the margin, which indicates the distance between the hyperplane and the nearest data points from each class. This makes it easier to differentiate between data categories.(Kausar et al., 2011).

SVMs prove highly valuable in analyzing intricate data that cannot be separated by a straightforward straight line. Nonlinear SVMs, also known as SVMs for nonlinear data, employ a mathematical technique that transforms data into higher-dimensional space, where identifying a boundary becomes more manageable.

SVM operates by establishing hyperplanes in multidimensional space for data that is linearly separable, utilizing a kernel function. These hyperplanes partition the sample data into various groups. The kernel function is instrumental in rearranging objects to achieve optimal separation. It essentially offers a way to transform input values x into a feature space denoted as $\phi(x)$ using a nonlinear mapping. While mapping from the input space to the feature space, the transformed entities can be distinguished by a linear boundary within the new space. (Ajdani & Ghaffary, 2021).

While SVM was originally developed to classify linearly separable data into two classes, it has evolved to handle datasets consisting of more than two classes that cannot be linearly separated. In multi-class scenarios, multiple binary SVMs are employed for classification. SVM demonstrates rapid training capabilities and excels in both classification and regression tasks.

Advantages of Support Vector Machine Algorithm

- ✓ It has a high level of accuracy
- ✓ Kernel SVM contains a non-linear transformation function to convert the complicated non-linearly separable data into linearly separable data
- ✓ It is effective on datasets that have multiple features
- ✓ It is effective when the number of features is greater than the number of data points
- ✓ It employs a subset of training points in the decision function or support vectors, making SVM memory efficient
- ✓ Apart from common kernels, it is also possible to specify custom kernels for the decision function

2.5.6 Data Preprocessing

Preprocessing stands as a pivotal stage in constructing a classification model for our research. Its purpose lies in transforming unwieldy and extensive data into a more refined and valuable form. The acquired data may contain missing values and categorical data, necessitating manipulation to ensure it attains an appropriate format for utilization by our models. The significance of preprocessing is underscored by its role in ensuring data consistency, which in turn has a direct impact on the model's accuracy. Once we've gathered and readied the dataset, the act of preprocessing, which involves cleaning the data prior to input into the classifiers, becomes imperative for the precise and accurate categorization of traffic. Therefore, preprocessing serves as an integral step in the overall process of finalizing our classification model. After the dataset is prepared, the subsequent pre-processing phase is carried out to provide a refined dataset to the machine-learning model.

2.6 Role of Machine Learning in the Intrusion Detection Sector

Machine learning plays a critical role in the intrusion detection sector by significantly enhancing the ability to identify and respond to cyber threats and attacks. In this context, intrusion detection refers to the continuous monitoring of network traffic and system activities to uncover unauthorized or malicious behaviors. Machine learning's strengths lie in its capacity to detect anomalies within large datasets, establish behavioral patterns, and learn from historical data. By analyzing deviations from established norms, machine learning models can swiftly pinpoint suspicious activities and generate alerts for potential intrusions in real time. This adaptive learning process empowers these models to evolve alongside emerging threats, effectively reducing false positives and improving the accuracy of threat detection(Haripriya & Jabbar, 2018).

Furthermore, machine learning enables the extraction of pertinent features from raw data, conducting multi-dimensional analyses that uncover intricate attack patterns, which might elude conventional rule-based systems. By integrating with threat intelligence sources, machine learning systems gain access to up-to-date information about known malicious entities and tactics, enhancing their capabilities. In some instances, they can even automate responses to detected intrusions, such as isolating compromised systems or implementing blocklists. In essence, machine learning's prowess in processing and comprehending vast volumes of data in real-time, adapting to new attack methodologies, and minimizing false positives make it an indispensable asset for fortifying the security and resilience of digital systems and networks against ever-evolving cyber threats(N. Kumari & Malhotra, 2022).

Support Vector Machines (SVM) play a significant role in intrusion detection systems (IDS) as a machine learning method employed to categorize network traffic or system behavior as either normal or potentially harmful. Intrusion detection revolves around the identification of unauthorized or suspicious actions that may pose a threat to network or system security. SVMs prove particularly advantageous in this context due to their capacity to manage data with numerous dimensions and intricate decision boundaries.

Following this, the SVM algorithm undergoes training to identify the most effective hyperplane that maximizes the separation between data points associated with normal behavior and those linked to intrusions. This robustness extends to handling noise and outliers effectively. Following training, the SVM can then classify new instances, initiating alerts or responses whenever an

intrusion is detected. Additionally, fine-tuning hyperparameters and experimenting with various kernel functions can further enhance performance.

2.7 Related Works

In this section, we examined and evaluated the research areas concerning DDoS attacks on normal networks. Additionally, we identified gaps and areas for improvement based on the scope of this thesis. It appears that many researchers who have explored the effects of DDoS attacks on networks have primarily concentrated on investigating the impact and detection of such attacks, overlooking the crucial aspect of preventing these attacks proactively. A majority of the papers reviewed did not incorporate any motivation or prevention mechanisms to ensure network accessibility for legitimate users during potential attacks.

Considering the inherent susceptibility of networks to security threats, numerous researchers engage in the examination of the impact of security attacks on network routing protocols, offering their insights into the detection of DDoS attacks. The majority of prior investigations have concentrated on the detection of low-rate denial of service attacks. They were passionate about finding how to detect attacks that are flowing slowly and in low-rate to the network. Most of the studies came up with novel algorithms that can detect the attack, but some of them detect only limited types of attacks, such as application layer attacks. In this part, recent malicious detection mechanisms designed by different authors are reviewed, and their gaps were presented in the following ways.

In the research work **Detection of DDoS Attacks Using Supervised Learning Technique (2020)** The paper identifies the motivations behind DDoS attacks, such as business feuds, cyber vandalism, extortion, cyber warfare, ideology, and data theft. The challenges associated with DDoS attacks include their dynamic and complex nature, making them difficult to detect and attribute accurately.

The paper introduces machine learning and its various types, with a focus on supervised learning techniques. It emphasizes the potential of supervised learning, including SVM, to detect DDoS attacks effectively. The proposed model aims to identify DDoS attacks in real-time by using SVM to classify network traffic patterns. The proposed model described in the paper involves the use

of a Honeypot detector and a proxy server to preprocess data. The paper emphasizes the advantage of rule creation to improve detection efficiency and security.

Additionally, the paper does not compare its results with existing DDoS detection methods or provide a thorough analysis of the model's performance metrics. Moreover, they tend to shift their focus to high volumetric DDoS attacks instead of following an inclusive approach for both low rate and high-rate DDoS attack(Prriyadarshini & Renuka Devi, 2021).

Power spectrum entropy-based detection and mitigation of low-rate dos attacks (2018) The paper focuses on addressing the challenge of detecting and mitigating Low-Rate Denial-of-Service (LDoS) attacks, which send periodic packet bursts to bottleneck routers, causing a degradation in TCP application performance. There are various security approaches to ensure different security approaches. The most important details in this article are the various types and solutions for Denial of Service (DDoS) attacks, which target availability. This research focuses mainly on low-rate DDoS attacks, which are a way to perform stealth DDoS attacks by slightly increasing data traffic from many nodes synchronously with respect to the nominal

The proposed detection model does not consider any further specification for the attacker, such as details of data (i.e., attacker can send different types of information) and the duration of attack. It also does not assume any specific traffic conditions so that the proposed approach is applicable to different conditions, such as one-way, two-way, high-velocity, and low-velocity. Along with addressing high-rate DDoS attacks, the suggested detection model not considered that there is a significant burden in terms of storage or computational resources required for revocation. (Chen et al., 2018).

Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques (2022) the authors address the growing threat of Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks on Internet of Things (IoT) systems. They propose a novel architecture consisting of two components: DoS/DDoS detection and DoS/DDoS mitigation. The detection component offers fine-grained detection by identifying the specific type of attack and the packet type used in the attack, allowing for corresponding mitigation countermeasures. The authors emphasize the need for more advanced detection methods beyond traditional signature-based intrusion detection systems to combat the evolving

nature of IoT-based attacks. However, the paper leaves a notable gap in terms of real-time detection of the attack and storage complexity of the model(Mihoub et al., 2022).

A Novel Approach for DDoS Attack Detection Using Big Data and Machine Learning (2021) The authors suggest an approach that relies on big data to detect DDoS attacks and recognize flash crowds. They utilize a dataset created through OMNET++ to train and test their machine learning model, and they use Apache Spark for efficient data processing. The effectiveness of their model is assessed using established statistical metrics. The paper also provides a review of related work, including adaptive density-based clustering techniques, K-means clustering, artificial neural networks, and statistical metrics for detecting and characterizing anomalies in DDoS attacks. In the proposed approach, routers extract incoming traffic characteristics and feed them into the machine learning model for classification. Malicious packets are identified and discarded, preventing the DDoS attacks from affecting the network. The preprocessing phase involves selecting a training dataset and utilizing Apache Spark for fast and scalable data preprocessing.

While this allows for controlled experiments, it may not fully capture the complexity and variability of real-world DDoS attacks. Volumetric DDoS attack such as TCP flooding are not well focused and low-rate DDoS attack where not covered(Gaurav et al., 2021).

Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning (2019) Volumetric DDoS attacks, responsible for over 65% of all DDoS incidents, flood targets with a high volume of useless data, causing network congestion and disruptions. The paper identifies the challenges of detecting and mitigating DDoS attacks effectively, despite various proposed solutions and guidelines. The ineffectiveness often arises from configuration errors and the absence of adaptive tools that can autonomously adapt to network dynamics. In response, the paper introduces "Smart Detection," a novel defense mechanism against DDoS attacks. This system classifies online traffic, distinguishing between high-volume and low-volume DDoS attacks, and operates as a sensor compatible with existing Internet infrastructure, avoiding the need for software or hardware upgrades. Smart Detection utilizes a Signature Dataset (SDS) and machine learning (ML) algorithms for detection, effectively addressing the challenges posed by both volumetric and low-volume DDoS attacks. The paper contributes by creating a signature database, addressing various DDoS attack types, and enabling automatic feature selection but

Storage complexity is not considered and the detection technique is offline(Lima Filho et al., 2019).

Table 1: Related Works

No.	Title	Problem addressed	Contributions/ achievements	Limitations
1	Detection of DDoS Attacks Using Supervised Learning Technique (2020)	High volumetric DDoS attack by using rule-based method	They arose up with a novel framework for detecting and mitigating volumetric DDoS attack	They tend to shift their focus to high volumetric DDoS attacks instead of following an inclusive approach for both low rate and high-rate DDoS attack.
2	Power spectrum entropy-based detection and mitigation of dos attacks (2018)	Detection and mitigation low-rate DDoS attacks.	A novel algorithm that uses power spectrum entropy to detect the low-rate DoS attack	Storage or computation overhead for revocation is high and high-rate DDoS attack where not focused.
3	Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques (2022)	Detection and mitigation of protocol level DDoS attacks.	Came up with new architecture “Looking Back” that includes two main components: DoS/DDoS detection and DoS/DDoS mitigation.	The paper focuses on offline detection and mitigation, however, real-time detection and immediate response to attacks are essential.
4	A Novel Approach for DDoS Attack Detection Using Big Data and Machine Learning (2021)	A big data-based method for the detection of DDoS attacks and flash crowds.	Minimizing UDP and ICMP floods DDoS attack.	Volumetric DDoS attack such as TCP flooding are not well focused and low-rate DDoS attack where not covered.
5	Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning (2019)	Identification various volumetric attacks, such as TCP flood, UDP flood, and HTTP flood.	They come up with Smart Detection of DoS and DDoS attack detection which uses traffic Signature along with ML.	Storage complexity is not considered and the detection technique is offline.

CHAPTER THREE

3. METHODOLOGY OF THE STUDY

3.1 Overview of Proposed methodology

The primary goal of this research is to employ machine learning techniques to identify and counteract DDoS attacks in real-time. Machine learning is currently highly regarded for its effectiveness in predicting intrusions and other occurrences, ultimately enhancing policy-making and reinforcing existing policies.

To accomplish these goals, the study will employ various methodologies and procedures. Additionally, an examination of current methods will aid in comprehending the suggested approach. This will include problem identification, proposing solutions, designing the implementation of the proposed solution, and testing various methods.

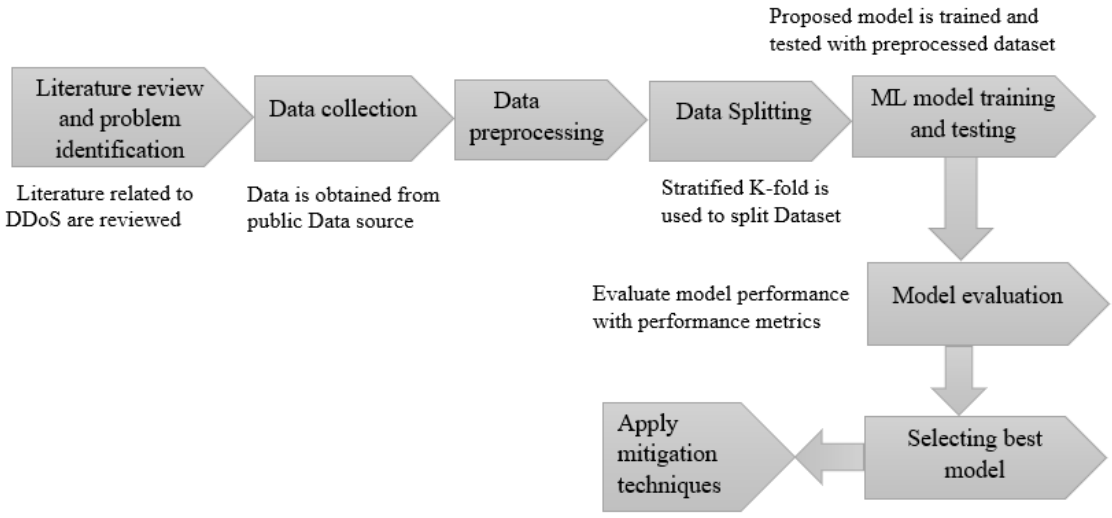

Figure 3: Research methodology

Literature Review

In this phase, different literature such as published journal articles, study papers, conference papers and reports related to security attacks in network were analyzed and paramount information that can enable to identify research problems are gathered.

Problem Identification

Based on the reviewed resources, the real-time detection and mitigation of denial-of-service attack has been identified as feasible research problems.

Design Proposed Solution

In this phase, the collected dataset will be prepressed and the machine learning models will be trained and tested. Finally, the model with best performance metrics will be selected and mitigation techniques will be applied on the traffic identified as attack by the best machine learning model.

We will employ a method that involves detecting attacked packets to train and evaluate our machine learning models. This technique distinguishes attacked packets from others by observing the rapid fluctuations in the position of the attacking node and the frequency of incoming packets, measured as broadcasts per second (broadcast/sec). Our machine learning models will receive training data consisting of both attacked and normal packets, one at a time.

3.2 DDoS Attack Classification Modelling

Handling Missing Values: Certain features within the dataset contain absent values, while others remain unfilled in terms of traffic data. It's not uncommon for traffic data to exhibit missing values, which play a crucial role in determining the potential for traffic to be classified as an attack. These omissions have a direct bearing on the accuracy of the classification model. Dealing with missing values within the dataset can be approached through various methods.

One approach involves the elimination of instances with missing values, provided that this action has a minimal impact on individual cases. Alternatively, the missing values can be filled by substituting them with zero values, which leaves the model unaffected. Additionally, missing datasets can be addressed through data imputation, a process that replaces the missing values with statistical measures like the mean, mode, or standard deviation, or by predicting these missing values based on available dataset information.

3.2.2 Feature Selection

In the realm of machine learning and data analysis, feature selection stands as a crucial procedure. It entails the careful selection of a subset of pertinent features (which can be thought of as variables, attributes, or columns) from the initial feature set. This endeavor serves multiple objectives, including the enhancement of model performance, reduction of computational intricacies, improvement of interpretability, and the mitigation of overfitting risks. In the context of network traffic data, encompassing DDoS datasets, feature selection methods typically fall into three principal categories: filter, wrapper, and embedded techniques(Torabi et al., n.d.).

3.2.2.1 Filter Methods

Filter methods assess the importance of features without considering the specific machine learning algorithm chosen. They rank features using statistical criteria like correlation, mutual information, or variance, and then pick a subset of features based on a predetermined threshold. Typical filter methods encompass:(Srivastava et al., 2013):

Correlation-based feature selection: Measures the correlation between each feature and the target variable.

Chi-squared test: Evaluates the independence between categorical features and the target.

Variance thresholding: Removes features with low variance, assuming they contain less useful information.

3.2.2.2 Wrapper Methods

Wrapper methods choose features by analyzing their influence on the performance of a particular machine learning algorithm. This process entails iteratively training and assessing the model using various feature subsets to identify the best set. Typical wrapper methods encompass:(Kumar, 2014):

Forward selection: Adds features one by one, starting with the most important, and evaluates their impact on model performance.

Backward elimination: Starts with all features and iteratively removes the least important ones.

Recursive Feature Elimination: Recursively removes the least significant features until a desired number is reached.

Bidirectional Feature Selection

Bidirectional feature selection is more effective when you have a relatively small dataset because it leverages the model's predictive power to guide feature selection. Filter methods, which rely solely on feature statistics (e.g., correlation, mutual information), might be limited in their effectiveness when the sample size is small.

Filter methods may not always identify the optimal set of features, but wrapper methods consistently offer the best subset of features.(Wang et al., 2016).

Machine Learning Feature Selection: To identify essential features for a given task, one can employ feature selection techniques rooted in machine learning, including ensemble methods. In this context, we have considered the Extra Trees Classifier method as a valuable tool. The feature selection techniques applied in this study leverage the Extra Trees Classifier. This classifier, utilizing the training dataset, generates multiple decision trees in a random manner.

Each decision tree is constructed by recursively splitting nodes based on a randomly selected subset of features. The Extra Trees Classifier further partitions the nodes within the decision tree by evaluating split quality using criteria such as entropy or the Gini index. These metrics serve to assess the quality of splits and aid in the selection of the most suitable attributes for data segmentation(Wang et al., 2016).

The entropy (E) is calculated using the formula:

$$Entroyp(E) = -\sum_{i=1}^{C}(pi * log2(pi))$$

where Σ_i denotes the sum over all possible classes (C), p_i represents the probability of class i, and $\log_2$ denotes the logarithm base 2.

Entropy is a measurement of the typical amount of data needed to determine the class label of a randomly selected element from a node. Using the distribution of class labels, it measures the degree of disorder or randomness in the node.

3.2.2.3 Embedded Methods

Embedded methods incorporate feature selection as part of the model training process itself. These methods optimize feature selection and model parameters simultaneously. Popular embedded methods include(Li et al., 2017):

Lasso Regression: Introduces L1 regularization to linear regression, encouraging sparsity and automatically selecting relevant features.

Tree-based methods: Decision Trees, Random Forests, and Gradient Boosting Machines can inherently measure feature importance during training and prune irrelevant features.

3.3 Classification Model Evaluations

The primary concern when constructing any machine learning model is to assess its effectiveness. The assessment of a classification model stands as a pivotal phase in the field of machine learning, allowing us to gauge how well the model is performing and whether it aligns with our desired benchmarks, encompassing accuracy, precision, recall, and other pertinent metrics.

Performance evaluation approaches can be categorized into two subtypes: Holdout and Cross-validation. Holdout evaluation offers an objective measure of learning performance by assessing a model using distinct data from its training dataset. In this method, the dataset is typically split randomly into a training set, validation set, and test set.

On the other hand, Cross-validation is a technique employed to evaluate model performance on a test dataset that the model has not encountered during training. This technique is particularly valuable in machine learning to evaluate the performance of a model on unseen data.(Al Lail et al., 2023).

3.3.1 Cross-Validation

Cross-validation is a machine learning evaluation method employed to evaluate the performance of a model on data it has not seen before. It entails dividing the accessible data into segments and using one segment as a validation set while training the model on the remaining data. This procedure is iterated with various segments serving as the validation set, and the outcomes are averaged to offer a more dependable performance assessment. The train-test-split method may result in the model missing some data due to repeated recordings during training, and it uses less

data for training and testing. So, for this reason we will use k-fold cross validation to spilt the dataset and evaluate our model performance(Rhohim et al., n.d.).

Cross-validation is a widely used technique for comparing and evaluating machine learning models. It divides data into training and testing portions, preventing overfitting and ensuring that models understand data patterns. We used the K-fold cross-validation method in our study to split the original dataset and assess the model's performance.

K-fold Cross-Validation is a method to determine machine learning model accuracy and practicality. It allows model comparison and assessment on unseen data. The process involves dividing the dataset into folds, using some for training and others for testing, and averaging the accuracy scores.

K-Fold Cross-Validation is used when dealing with imbalanced data distributions(Shrivas & Mishra, n.d.). It prevents biased model training by ensuring each fold's class distribution matches the overall dataset, addressing the issue of unbalanced classes.

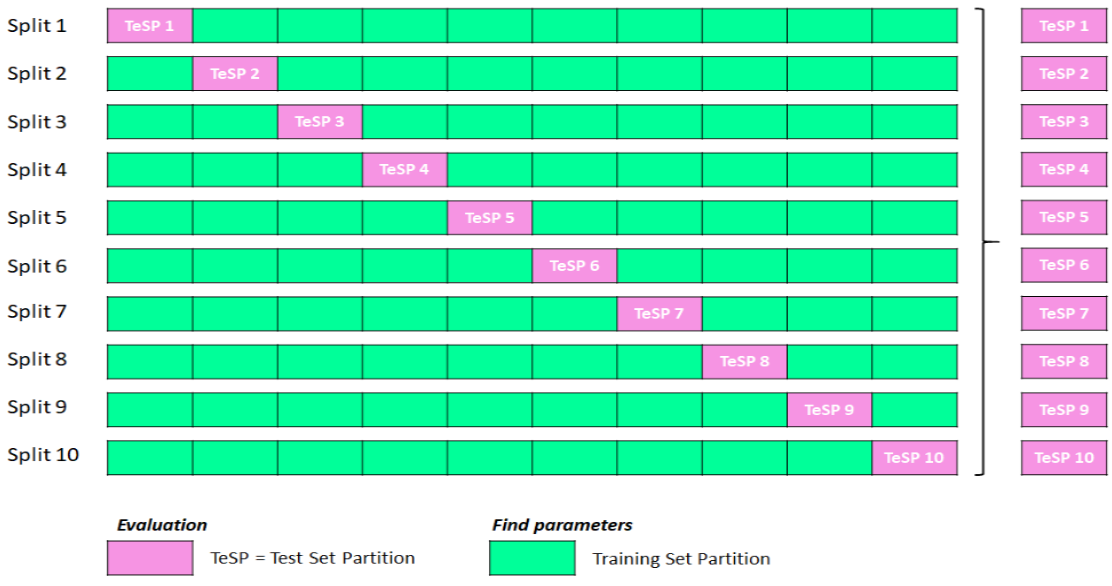

Figure 4: Research methodology

3.3.2 Classification Model Performance Evaluation Metrics

The idea behind employing a machine-learning algorithm for constructing a classification model revolves around the notion of receiving valuable feedback. Utilizing a machine-learning algorithm, you generate the model, obtain input regarding evaluation criteria, adjust, and iterate until achieving the desired level of accuracy. Metrics designed for gauging performance play a pivotal role in assessing the model's effectiveness. The capacity of evaluation metrics to differentiate between the outcomes of different models constitutes a fundamental characteristic. In this research, multiple metrics, such as accuracy, precision, recall, the f1-score, and confusion matrices, are utilized to evaluate the effectiveness of the chosen classification model. Terms like True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are frequently used in multiclass classification, along with confusion matrices, to assess the performance of a machine learning model.(M & M.N, 2015).

When constructing a classification model through a machine learning algorithm, the fundamental concept centers on obtaining useful feedback. This process entails utilizing the algorithm to construct the model, seeking input on assessment criteria, making necessary adjustments, and iterating until the desired level of accuracy is attained. Performance evaluation metrics play a pivotal role in assessing the model's effectiveness. One critical aspect of these metrics is their capacity to differentiate between the outcomes of various models. In this research, a variety of metrics, including accuracy, precision, recall, the F1-score, and confusion matrices, are employed to measure the efficacy of the chosen classification model. Furthermore, terms like True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are commonly utilized in multiclass classification and confusion matrices to assess the performance of the machine learning model.

- **True Positive (TP)**

A True Positive outcome signifies the accurate recognition of a danger or weakness by a system or tool. This implies that the system has adeptly identified and confirmed the presence of a possible threat or vulnerability, subsequently implementing the necessary measures to address or resolve it. To illustrate, when an intrusion detection system (IDS) accurately identifies and reports an attempted cyber-attack, it qualifies as a True Positive. Such outcomes are of paramount importance

in upholding the safety and security of computer systems and networks(*True Positive Rate Definition | Encord*, n.d.).

- **True Negative (TN)**

A True Negative outcome signifies an accurate verdict made by a system or tool that no threat or vulnerability is present. In essence, the system has concluded that there is no impending threat or vulnerability and has refrained from taking any action. For instance, if antivirus software determines that a file is free from any virus infection, and indeed it is clean, this constitutes a True Negative result. This type of outcome holds equal significance to True Positives, as it serves to prevent unnecessary actions and alerts that might otherwise cause confusion or unnecessary panic(*Classification: True vs. False and Positive vs. Negative | Machine Learning | Google for Developers*, n.d.).

- **False Positive (FP)**

A False Positive outcome indicates an instance where a system or tool falls short in recognizing a threat or vulnerability. In essence, the system has overlooked a potential threat or vulnerability and has refrained from taking any action to address it. For instance, if antivirus software fails to spot a virus in a file that is, in fact, infected, this would be categorized as a False Positive. Such outcomes carry risks as they permit threats and vulnerabilities to remain undetected, potentially leading to harm to computer systems and networks(*What Is False Positive Rate,* n.d.).

- **False Negatives (FN)**

A False Negative outcome occurs when a system or tool mistakenly fails to recognize a threat or vulnerability. In this scenario, the system erroneously detects something as harmless when it could pose a genuine risk, leading to unnecessary actions being taken. For instance, if an Intrusion Detection System (IDS) wrongly categorizes valid network traffic as a cyberattack attempt, it would be classified as a False Negative. Such results can be vexing for users and result in the inefficient utilization of time and resources(Evaluating the Performance of Machine Learning Models | by Frank Liang | Towards Data Science, n.d.).

3.3.2.1 Accuracy

The Accuracy (ACC) metric is among the most straightforward classification metrics to use, and it's calculated by dividing the number of correct classifications by the total number of classes. It quantifies how accurately the model classifies the test dataset, and it's obtained by dividing the total number of correct classifications by the number of instances that were classified correctly.(Sweeney et al., 2022).

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

3.3.2.2 Precision

The precision metric serves as a remedy for the limitation of Accuracy. Precision measures the percentage of positive classifications that were accurately identified. This can be computed by taking the True Positive, which are correctly identified positives, divided by the total positive classifications (sum of True Positive and False Positive)(Michaud et al., 2022).

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

3.3.2.3 Recall

This metric, similar to Precision, focuses on quantifying the ratio of genuine positives that were identified erroneously. It can be computed as the True Positive, which represents correct positive identifications, divided by the total number of actual positives, encompassing those correctly identified as positive as well as those inaccurately identified as negative (True Positive and False Negative)(Martin Ward Powers, 2011). $Recall = \frac{TP}{TP+FN}$

3.3.2.4 F1- Score

The F-score or F1 Score is a metric used to assess a Multiclass classification model's performance concerning positive class classifications. It is determined using Precision and Recall and serves as a unified measure encapsulating both Precision and Recall. Consequently, the F1 Score can be computed as the harmonic mean of both Precision and Recall, giving equal importance to each of them(Sokolova et al., 2006). $F1\_Score = 2 * ((\frac{Precision*Recall}{Precision+Recall}))$

3.3.2.5 Confusion Matrix

A confusion matrix is a structured display of classification results from a multiclass classifier, employed to depict how well the classification model performs on a dataset with known true values. It is a table that summarizes the performance of the classification model, indicating the quantities of true positives, true negatives, false positives, and false negatives(Karimi, n.d.).

CHAPTER FOUR

4. PROPOSED REAL-TIME DETECTION and MITIGATION of DDOS ATTACK

This chapter delves into the general methodology and necessary steps for creating a machine learning-based classification model designed to identify DDoS attacks based on specific distinguishing attributes. In the field of cybersecurity, machine learning plays a critical role in managing vast volumes of data that can pose challenges for human analysis. To build these classification machine learning models, a variety of methods, techniques, and procedures must be applied throughout each stage of the development process.

There exist multiple techniques for preventing, detecting, and mitigating DDoS attacks. In terms of detection methods, two primary approaches are recognized: signature-based and anomaly detection. Signature-based methods can only recognize attacks with known signatures, making them ineffective against new or zero-day attacks. In contrast, the anomaly detection approach can detect novel and unfamiliar attacks by identifying unusual patterns or circumstances resulting from the attack. Statistical techniques like entropy analysis and machine learning (ML) methods are typically employed in the anomaly detection approach.

4.1 Proposed DDoS Attack Detection and Mitigation Model Architecture

Creating a robust model architecture for detecting and countering Distributed Denial of Service (DDoS) attacks involves a blend of techniques and components to effectively identify and combat malicious traffic.

The proposed solution consists of two primary phases: Detection and Mitigation. Initially, data is collected from CICIDS 2019 Dataset sources available on Kaggle, and various data preprocessing methods such as data integration, transformation, reduction, and feature selection are applied. Subsequently, the preprocessed data is divided into training and testing with k-fold cross validation technique. The training set is utilized to train a machine learning model to recognize both attack and normal traffic patterns, while the test set is employed to assess the model's performance. Once

the dataset is partitioned, the machine learning model is trained to grasp the data's underlying patterns.

Next, the model is tested to determine if it can successfully identify and understand network traffic behavior. Ultimately, any traffic identified as a DDoS attack is blocked from the network server.

In summary, during the detection phase, traffic patterns are analyzed, and the traffic is categorized as either DDoS attacks or normal traffic. In the mitigation phase, any traffic classified as a DDoS attack is actively removed from the network.

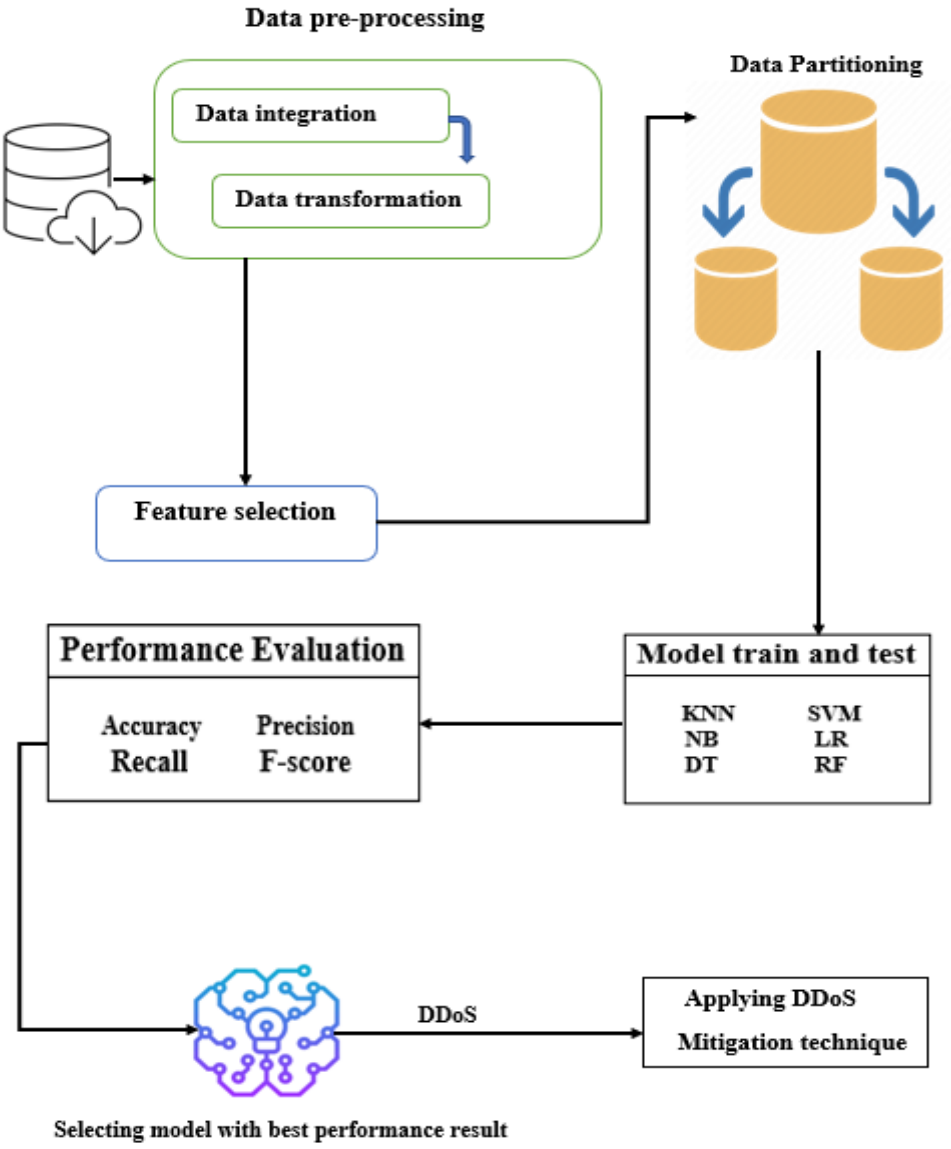

Figure 5: Proposed DDoS classification and mitigation model architecture

4.2 Understanding of the Data

Once the problem is clearly defined and dataset is obtained from CICIDS 2019 Dataset sources available on Kaggle, the subsequent phase involves delving into the realm of data understanding, wherein the aim is to identify the datasets that are within reach. The effectiveness of machine learning and knowledge discovery greatly depends on the quality and quantity of available datasets. In this procedure, the datasets are thoroughly inspected to ensure they are complete, free of duplicates, missing values, and that the feature values are plausible, along with other pertinent criteria. Here, below is the description of dataset features.

Note: The data type for all features is numeric

- Source.IP and Destination.IP: These features help identify the source and destination of network traffic, which is essential for tracking communication patterns and identifying potential malicious sources.
- Idle_Max and Idle_Mean: These features capture the idle times between packets in a flow. Sudden spikes or irregularities in these values can indicate the presence of DDoS attacks, as attackers might try to overwhelm the network with bursts of traffic.
- Flow_IAT_Mean, Flow_IAT_Max, Flow_IAT_Std: These features provide insights into the inter-arrival times between packets. Unusual patterns in inter-arrival times might suggest a coordinated attack or botnet activity.
- Fwd_IAT_Total, Fwd_IAT_Mean, Fwd_IAT_Std, Fwd_IAT_Max: These features focus on the inter-arrival times in the forward direction of a flow. DDoS attacks often involve massive amounts of traffic from multiple sources. Unusual inter-arrival times can indicate coordinated attacks or traffic anomalies.
- Fwd_Packet_Length_Std, Fwd_Packet_Length_Max, Fwd_Packet_Length_Mean: These features provide insights into the characteristics of packet lengths in the forward direction. Drastic variations in packet lengths can be indicative of DDoS attacks or other anomalies.
- Fwd_Packets_Sec, Flow_Bytes_Sec, Flow_Packets_Sec: These features measure the rates of packet and byte transfers. Rapid increases in these rates might signal DDoS attacks attempting to flood the network.

- Subflow_Fwd_Bytes, Fwd_Header_Length, Avg_Fwd_Segment_Size: These features offer further information about the data being sent in the forward direction. DDoS attacks often involve unusual traffic patterns that can be detected by analyzing these metrics.
- Packet_Length_Variance, Packet_Length_Std, Packet_Length_Mean: These features provide statistics about packet lengths within a flow. Changes in these values could indicate malicious activity, as attackers might send packets of unusual sizes.
- Max_Packet_Length, Average_Packet_Size: These features provide additional insights into the distribution of packet sizes, which can help identify anomalies caused by DDoS attacks.
- Init_Win_bytes_backward, min_seg_size_forward: These features focus on characteristics of the TCP protocol. Abnormal values could be indicative of malicious activity targeting network resources.
- Bwd_Packets_Sec: This feature measures the rate of backward (opposite direction) packets. Unusual behavior in the reverse flow can be a sign of certain types of DDoS attacks.
- Packet_Length_Variance: This feature represents the statistical variance of packet lengths within a flow. Anomalous variance might indicate the presence of DDoS attacks where packets are of varying and unusual sizes.
- Packet_Length_Std: Similar to variance, this feature represents the standard deviation of packet lengths within a flow. High standard deviation could be a sign of irregular packet sizes associated with attacks.
- Packet_Length_Mean: This feature provides the mean (average) packet length within a flow. Sudden changes in average packet size can indicate potential attacks.
- Flow_Bytes_Sec: This feature measures the rate of bytes transferred per second within a flow. Rapid increases in this rate might be indicative of DDoS attacks that aim to flood the network with excessive data.
- Flow_Packets_Sec: This feature measures the rate of packets transferred per second within a flow. Abnormally high rates might indicate DDoS attacks with a high packet volume.
- Subflow_Fwd_Bytes: This feature represents the number of bytes transferred in the forward direction of a subflow. Unusual values could indicate attacks involving segmented traffic.

- Fwd_Header_Length: This feature represents the length of headers in the forward direction of a flow. Changes in header length might be indicative of attack traffic attempting to bypass network defenses.
- Avg_Fwd_Segment_Size: This feature provides the average size of segments in the forward direction of a flow. Deviations from normal segment sizes could be associated with DDoS attacks.
- Max_Packet_Length: This feature indicates the maximum length of a packet within a flow. Large packet lengths might be part of an attack strategy.
- Average_Packet_Size: Similar to average packet length, this feature gives the average size of packets within a flow. Sudden changes in this value can be indicative of attack traffic.
- Init_Win_bytes_backward: This feature represents the initial window size in the backward direction of a flow. Unusual values could indicate attack attempts that exploit TCP window size settings.
- min_seg_size_forward: This feature represents the minimum segment size in the forward direction of a flow. Deviations from typical segment sizes might suggest attack traffic.
- Bwd_Packets_Sec: This feature measures the rate of backward (opposite direction) packets per second. Unusual activity in the reverse flow could indicate certain types of DDoS attacks.

4.3 Preprocessing Implementations

To put the recommended solution into action in this research, a range of tools and software packages have been employed. Throughout the entire process, from data preprocessing to the ultimate step of constructing the model, Python has been chosen as the programming language for the implementation and experimentation of each proposed solution. Below, you'll find an overview of the development tools utilized in this research project.

Table 2: Development tools

Tools and Packages	Version	Description
Anaconda Navigator	2.4.2	Assist us in swiftly and conveniently initiating development applications. Administer Anaconda environments, channels, and packages without the necessity of using command-line utilities.
Python	3.10.12	Python is a well-liked, simple-to-learn programming language that can be used to create machine learning applications.
Scikit-Learn	1.2.2	This Python module is employed in machine learning to manage fundamental tasks such as clustering, regression, and classification.
Pandas	1.5.3	Working with "relational" or "labeled" data can be made easier by using Pandas, a significant package that offers quick, adaptable, and expressive data structures. Incorporating, filtering, and loading data from other sources are all made possible. Handling the dataset and using external sources like Excel.
NumPy	1.22.4	This study utilizes arrays for processing numbers, strings, and objects. It employs these arrays for tasks like converting text to numeric data, training models, and conducting model testing.
NetFlow	6.5.3	It is a network protocol system created by Cisco that collects active IP network traffic as it flows in or out of an interface.
Seaborn	0.12.2	Statistical data visualization
Matplotlib	3.7.1	Python figures of publication quality. It is utilized in this study for the visualization of data and findings.

Joblib	1.2.0	Joblib is made to handle large amounts of data simply and effectively. For persistent Python objects, it offers a simple interface that can be helpful for distributed computing or lengthy processes.
Flaskserver	1.1.1	This research is using a small python framework to create web services, and they're using it to build a prototype for some chosen models.
Microsoft Excel	2019	For dataset preparation
Microsoft Word	2019	For documentation purpose

The preprocessing procedure encompasses tasks such as addressing missing values and selecting features, all of which are carried out using the Python programming language.

```
import pandas as pd
import numpy as np
```

Figure 6: Implementation Importing pandas' library

4.3.1 Data Integration

During this phase, data from diverse origins has been consolidated. Given that online datasets are typically designed for specific purposes and may not encompass all the necessary information for both identifying and mitigating DDoS attacks, some essential details were missing from the primary dataset obtained from Kaggle. In order to apply mitigation technique, it is important to have Ip addresses. Since the data set we obtained from CICIDS 2019 Dataset sources does not contains the features we needed to mitigate the incoming attack, we need to generate random IP address or we need to obtain the features from other sources having similar patterns of attack. To address this gap, we sought out source and destination IP addresses from DDoS Dataset sources on Kaggle and integrated them with the initial dataset.

4.3.2 Data Transformation

This stage deals with converting the data into a format that is suitable for the machine learning algorithm. In this regard, since both source and destination IP addresses are in String format, we cannot train them with other variables so we need to convert them to the format that is number to train them with other variables, and this is done.

4.3.3 Dimensionality Reduction

In this stage, the number of variables is reduced based on their significance level for determining the network traffic pattern as either it attacks or it is normal network traffic. In relation to this, the number of features is reduced from 35 to 28 according to their significance and classification levels.

4.3.4 Handling Missing Value Implementation

To handle missing data in the dataset, we made use of the fillna() method, which substitutes these gaps by calculating the average and most frequent values. For this task, we utilized the Python Simple Imputer module, applying the mean strategy to replace the missing values.

```
DatasetUrl = 'https://raw.githubusercontent.com/GutemaB2015/DDOS-Detection-and-Mitigation/main/FullDataset.csv'
data = pd.read_csv(DatasetUrl)
label_encoder = preprocessing.LabelEncoder()
data['Label'] = label_encoder.fit_transform(data['Label'])
data['Label'].fillna(data['Label'].median(), inplace=True)
```

Figure 7: Implementation replacing missing value

After cleaning the dataset, we separated the features into independent and dependent ones. The data, without any feature selection, was then assigned as X training data, while the labels, which include the class or label dataset, were designated as Y. We proceeded to train the classifiers using the entire dataset by using the fit() method, initializing each classifier with the appropriate parameter settings.

4.4 Machine Learning Models Implementation

We employed the Python scikit-learn library package to build machine learning models utilizing the proposed algorithm. We adhered to the standard procedure of importing the necessary library packages for both modeling and assessing the models. To achieve this, we imported all the required Python libraries as outlined below.

```
#importing necessary libraries.
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
from sklearn.metrics import precision_recall_fscore_support
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn import metrics
# DatasetUrl
DatasetUrl = 'https://raw.githubusercontent.com/GutemaB2015/DDOS-Detection-and-Mitigation/main/FullDataset.csv'
data = pd.read_csv(DatasetUrl)
```

Figure 8: Implementation Importing necessary libraries

4.5 Real-time Data Processing Implementation

Apache Kafka, often referred to simply as Kafka, is an open-source, distributed streaming platform designed for real-time applications. Among its many capabilities, Kafka empowers the creation of event-driven applications that operate in real-time. In our specific scenario, we leverage Kafka to capture the continuous flow of data in real-time and subject it to analysis through our machine learning model. To begin this process, our first step involves crafting a consumer application using the Kafka-python library in Python. This consumer, which we can think of as a server, plays a pivotal role in retrieving Kafka messages while considering the offsets. These offsets are unique identifiers assigned to each message and are associated with the source IP address of the incoming network traffic.

```
from kafka import KafkaConsumer, TopicPartition
from concurrent.futures import ThreadPoolExecutor
sample_consumer = KafkaConsumer(
 bootstrap_servers=hosts,
 group_id=group_id,
 auto_offset_reset=auto_offset_reset,
 enable_auto_commit=False,
 session_timeout_ms=session_timeout_ms,
)

def process_job(kafka_message,topic_offset):
    # commit the offset
    consumer.commit(offsets=offsets)

for kafka_message in consumer:
    # get partition and offset of message
    topic_partition = TopicPartition(topic_name, kafka_message.partition)
    offset = OffsetAndMetadata(kafka_message.offset + 1, None)
    topic_offset = {topic_partition: offset}

    executor = ThreadPoolExecutor(max_workers=number_of_threads)
    executor.submit(process_job, kafka_message,topic_offset)
```

Figure 9: Implementation of Kafka to capture real-time traffic flow

In general, Kafka are committed once the consumer (Server) gets the packet from the attacker. “python-kafka” also gives us the flexibility to commit whenever required. As soon as the attack packet arrives at target server, the Kafka initiated and start capturing the traffic from the source.

4.6 Hyperparameter Tuning Implementation

Hyperparameter tuning refers to the task of choosing the optimal configuration of hyperparameters for a machine learning model. Hyperparameters are parameters that are predefined and not learned from the training data. They play a significant role in governing different aspects of the training process and the model's complexity, thereby impacting how the model learns and generalizes from the data.

```
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

# Define the Bidirectional Feature Selection (BFS)
bfs_selector = SelectKBest(score_func=f_classif, k='all')
X_train_selected = bfs_selector.fit_transform(X_train, y_train)
X_test_selected = bfs_selector.transform(X_test)

# Define the SVM hyperparameter grid
svm_param_grid = {
    'C': [0.1, 450, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto', 0.1]
}

# Create an SVM classifier
svm_classifier = SVC()
svm_grid_search = GridSearchCV(svm_classifier, svm_param_grid,
cv=KFold(n_splits=10, shuffle=True, random_state=42), verbose=2, n_jobs=-1
svm_grid_search.fit(X_train_selected, y_train)

# Get the best SVM model
best_svm_classifier = svm_grid_search.best_estimator_

# Evaluate the SVM model
y_pred = best_svm_classifier.predict(X_test_selected)
print('SVM Classifier Accuracy:', metrics.accuracy_score(y_test, y_pred))
```

Figure 10: Implementation Hyperparameter Tuning Implementation

4.7 Implementation of Mitigation mechanism

An IP blocking mechanism plays a pivotal role in the mitigation of Distributed Denial of Service (DDoS) attacks(Dayanandam et al., 2019) as explained on reviewed literature. Since DDoS attacks are malicious efforts to incapacitate a target system, network, or service by inundating it with a

deluge of traffic. This mechanism serves as a crucial line of defense by identifying and intercepting traffic originating from malicious IP addresses(Zulkii Hasan et al., 2023).

The process starts while machine learning model stars to identify traffic as DDoS, their respective source IP address start to be captured and added to the array of blocked Ip address list. Once the Ip address is on list of blocked Ip address, the attacker will be blocked and cannot access the network with the same Ip address again.

Moreover, the implementation below shows how attackers Ip address is blocked.

```
# Create an SVM classifier
svmClassifier = SVC()
param_grid = {
'C': [0.1, 450, 10],
'kernel': ['linear', 'rbf', 'poly'],
'gamma': [0.1, 'scale', 'auto']}
svmClassifier.fit(X_train, y_train)
y_pred = svmClassifier.predict(X_test)

prediction = svmClassifier.predict(requestTraffic)

if prediction == 0:
  print("The traffic is not a DDoS attack.")
else:
  # Create a list of blocked IP addresses
  blocked_ips = []

  if requestTraffic not in blocked_ips:
    # Add the DDOS traffic request to the blocked list
    blocked_ips.append(requestTraffic)
    print("a DDoS attack Blocked.")

    with open("blocked_ips.txt", "w") as f:
      for requestTraffic in blocked_ips:
        f.write(str(requestTraffic )+ "\n")
  else:
     print("This request is already blocked.")
```

Figure 11: Sample code for Implementation of Ip blocking

```
NetFlow Packet :  [24.093229902889156, 28.578797954655844, 89.18050926878858, 88.34841266940653,
A DDoS attack is detected and blocked.
NetFlow Packet :  [56.95278751946087, 71.46731193208733, 39.39385851535951, 51.26155221636248,
A DDoS attack is detected and blocked.
NetFlow Packet :  [56.95278751946087, 71.46731193208733, 39.39385851535951, 51.26155221636248,
A DDoS attack is detected and blocked.
NetFlow Packet :  [48.75954860505923, 48.04505617199709, 14.233573815099565, 20.09350006501037,
A DDoS attack is detected and blocked.
```

Figure 12: Implementation for mitigation of DDoS packet

4.8 Model Evaluation Implementation

After completing data preprocessing, feature selection, and model building, the next step involves evaluating the machine learning model's performance. This assessment is carried out through a stratified k-fold cross-validation approach with k set to 10, along with various performance metrics like accuracy, confusion matrix, precision, recall, and f1 score. The implementation of stratified k-fold cross-validation is achieved using the cross_val_score () and cross_val_predict () methods. The cross_val_score () method is part of the Sklearn Python model selection package and takes parameters such as estimator, features, target, fold count, scoring method, and n_jobs to specify the number of CPUs for computation.

In this research, the entire dataset undergoes a 10-fold stratified cross-validation, where n_jobs indicate the number of CPUs used for computation. A setting of n_jobs equal to -1 utilizes all available CPUs for computation, while a value of 1 indicates the opposite. Additionally, the cross_val_predict () method provides predicted target values for the utilized fold. Python modules from the sklearn library, including Classification report () for generating a comprehensive report with essential ranking metrics, precision_score () for computing the model's accuracy score, and confusion_matrix () for evaluating classification accuracy, are applied in this study.

```
svm_model = SVC()

# Create the GridSearchCV object
grid_search = GridSearchCV(svm_model, param_grid, cv=10)

# Fit the model to the data
grid_search.fit(X_train, y_train)

y_pred = grid_search.predict(X_test)
print("SVM Classifier Accuracy:", grid_search.score(X_test, y_test))

detailed_results = grid_search.grid_scores_
for result in detailed_results:
    print("Parameters:", result.parameters)
    print("Mean Test Score:", result.mean_validation_score)
    print("All Test Scores:", result.cv_validation_scores)
```

Figure 13: Implementation creating object for svc

CHAPTER FIVE

5. RESULTS EVALUATION, and DISCUSSIONS

The results presented in this section highlight the success of our proposed real-time DDoS detection and mitigation Model in the context of machine learning. The high accuracy, precision, and recall achieved by the Support Vector Machines algorithm demonstrate its ability to accurately identify DDoS attacks, contributing to enhanced network security.

The main aim of this chapter is to present the details of the experimental results obtained from this study. The overview model performance, evaluation results, and the effect of feature selection on performance are discussed in this chapter.

5.1 Dataset Class Distributions Results

The total dataset size used for this study is 3000 rows with 28 columns including one target class after preprocessing the row data collected from. 459 traffic are DDoS slowloris traffic, 1184 DDoS traffic and 1356 with BENIGN traffic. Class distribution for three class datasets before balancing are shown in detail in figure 12 below

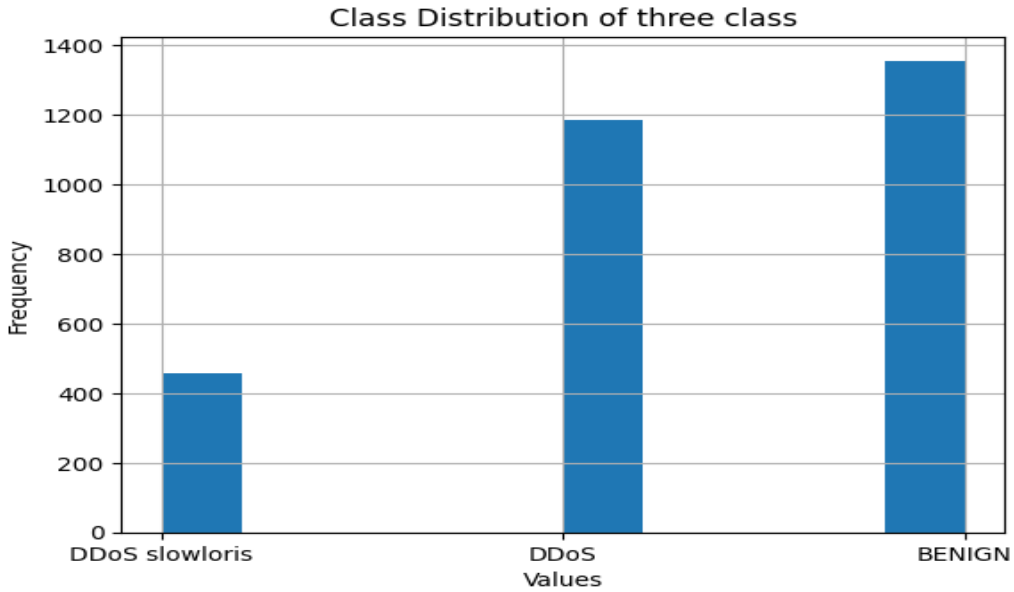

Figure 13: Class distribution for three class before data balancing

Initially, our dataset consisted of 3,000 rows and 28 columns, making it substantial and feature-rich. However, the distribution of classes within this dataset was significantly imbalanced. Out of these 3,000 instances, we had 459 cases of DDoS Slowloris traffic, 1,184 instances of DDoS traffic, and a predominant 1,356 instances labeled as BENIGN traffic. This class imbalance could potentially lead to skewed model performance, with the model being more inclined to classify new data points as BENIGN due to its overwhelming presence in the original dataset.

To rectify this imbalance issue, we employed Synthetic Minority Over-sampling Technique (SMOTE). SMOTE is a powerful method for rebalancing datasets, especially when dealing with imbalanced classes. Class distribution for three class datasets after balancing are shown in detail in figure 13 below

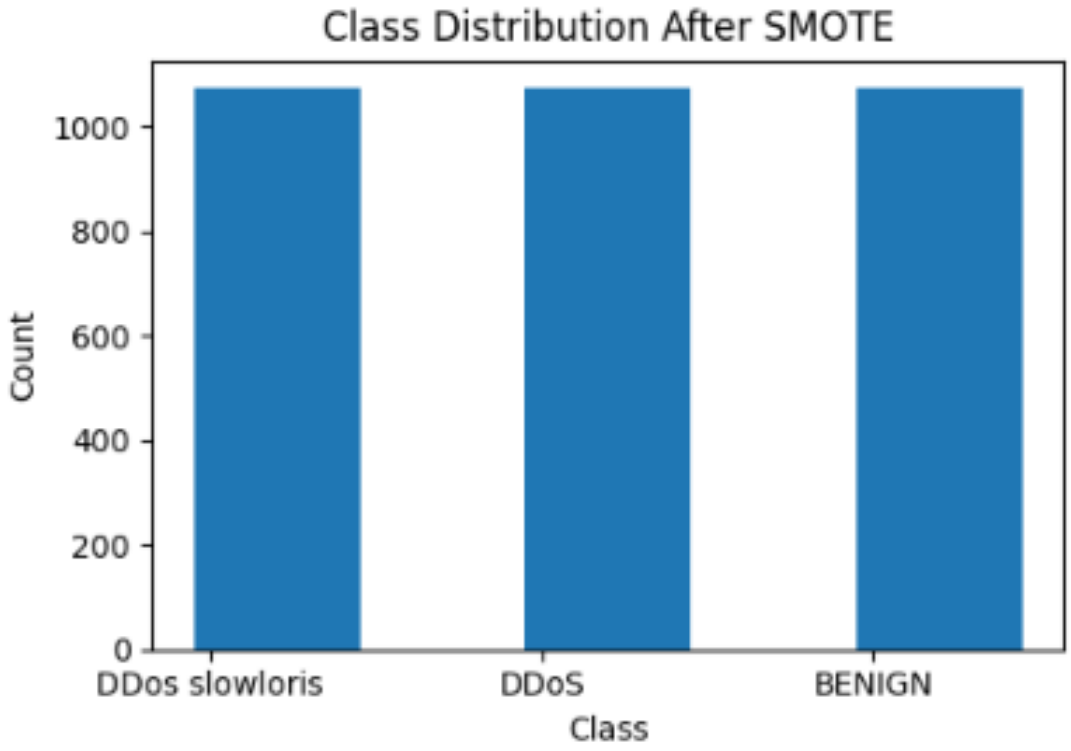

Figure 14: Class distributions for three class after data balancing

Now, each class i.e. DDoS Slowloris, DDoS traffic, and BENIGN contains an equal and balanced representation of relatively 1,000 instances excluding label. This balance was achieved by generating synthetic samples for the minority classes (DDoS Slowloris and DDoS traffic) to match the size of the majority class (BENIGN). These synthetic instances closely resemble the original instances but introduce slight variations to diversify the dataset.

This balanced dataset is a crucial enhancement for machine learning models, particularly in the context of intrusion detection or any application where classifying different types of network traffic accurately is essential for cybersecurity. With this new dataset, our models are less likely to be biased towards the majority class, enabling them to make more accurate and fair predictions across all three traffic categories i.e. DDoS Slowloris, DDoS traffic, and BENIGN.

5.2. Feature importance and Selection Result

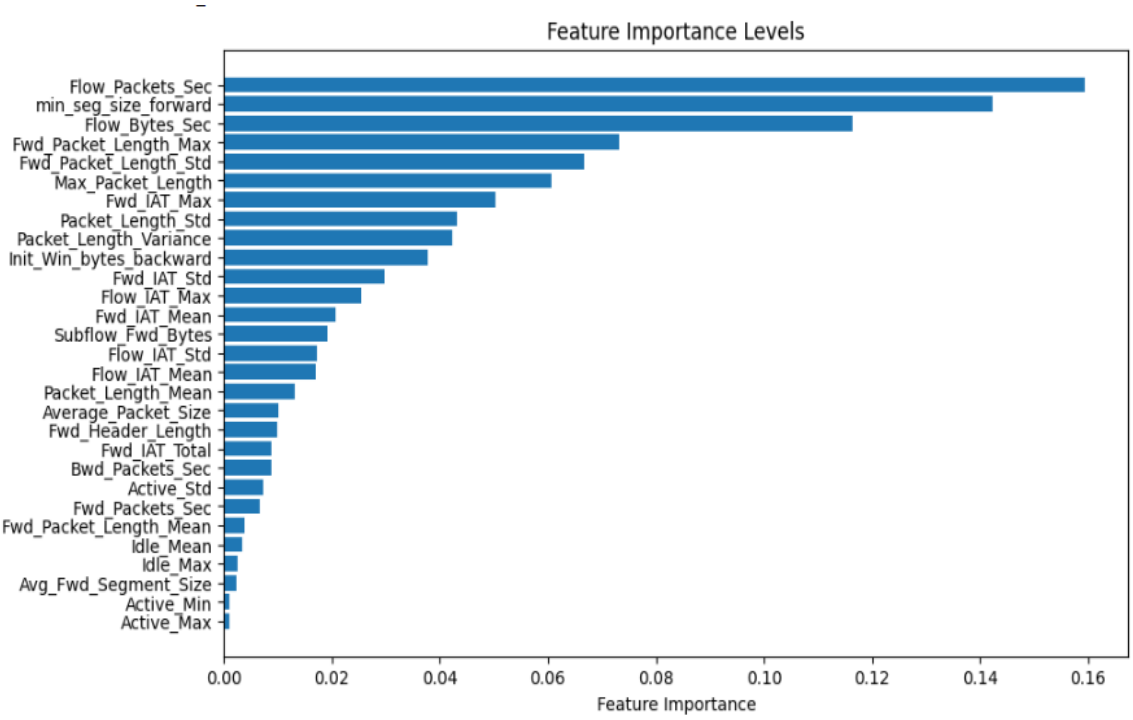

Figure 15: Feature Importance level

Figure 15 illustrates the feature ranking based on their relevance scores using the random forest classifier algorithm. The importance of features signifies their significance in determining the model's target accuracy. Features with higher importance scores exhibit better performance compared to those with lower scores. The comparison among features is based on evaluating the importance scores relative to other features in the dataset. In this study, feature selection methods are applied to choose the most suitable feature set, enhancing the discriminative power of machine learning classifiers. Bidirectional feature selection with cross-validation is employed during the training of SVM, RF, DT, LR, NB, and KNN models using the training dataset. The table below displays the results of feature selection for each of the selected algorithms.

Table 3: Feature selection result

Feature Selection Method	Selected ML model	Selected Feature	Original Feature
Bidirectional Feature Selection	SVM	28	35
	DT	28	35
	RF	28	35
	NB	28	35
	KNN	28	35
	LR	28	35

5.3 Models Performance and Evaluation Result

The experiment developed a Support Vector Machine (SVM) classifier algorithm designed to categorize network traffic into three distinct classes: DDoS, DDoS slowloris, or Benign. Testing and evaluating the model's performance were conducted using a 10-fold cross-validation approach, as explained in earlier chapters (three and four). The dataset was divided into ten subsets, and the model underwent ten iterations of training and evaluation, with each iteration employing a different subset as the test set. Below, we present the results showcasing the performance of our model.

5.3.1 Confusion Matrix for SVM

A confusion matrix is a table used in machine learning to evaluate the performance of a classification model, including SVM (Support Vector Machine). It summarizes the model's predictions compared to the actual ground truth.

As depicted in Figure 16 below, the SVM algorithm's performance is evaluated on a network traffic dataset using a confusion matrix. Specifically, 281 out of 289 instances are accurately classified as DDoS slowloris, 258 out of 279 instances as DDoS, and 255 out of 256 instances as Benign. However, there are some misclassifications where the algorithm incorrectly categorizes 8 instances of DDoS slowloris as DDoS and Benign, 11 instances of DDoS as DDoS slowloris and Benign, and 1 instance of Benign as DDoS.

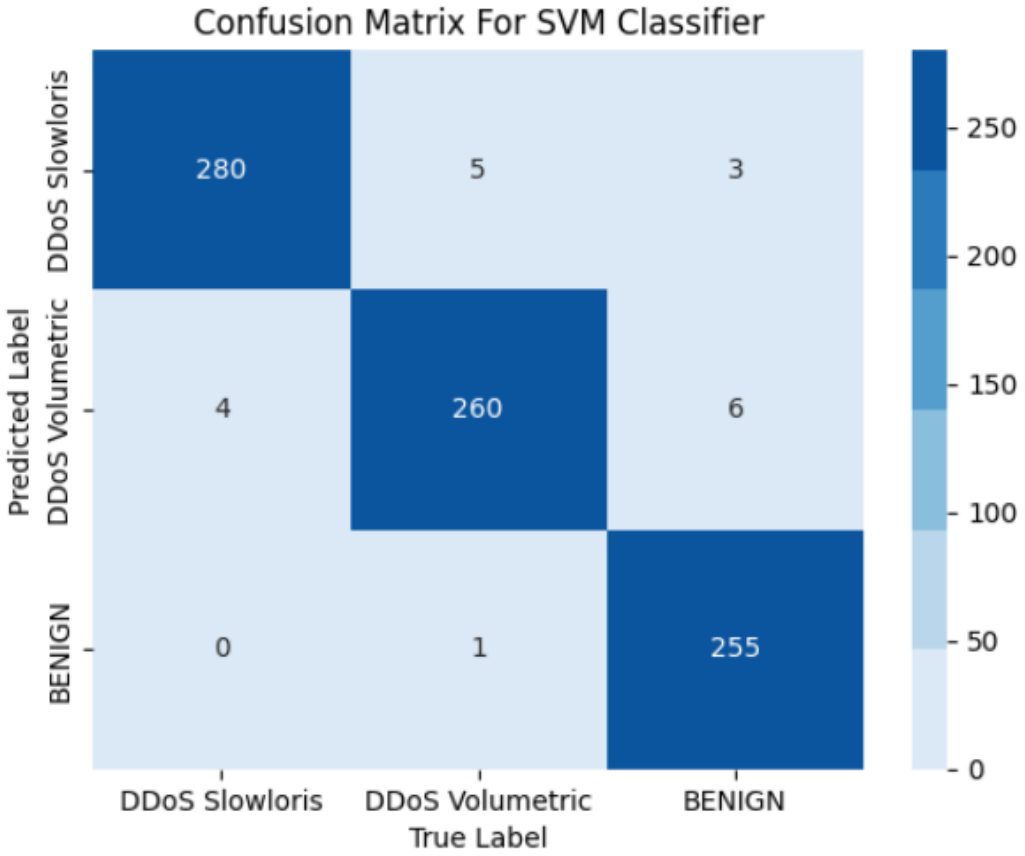

Figure 16: Confusion Matrix for SVM

5.3.2 Confusion Matrix for Logistic Regression Classifier

As depicted in Figure 17 below, the LR algorithm's performance is assessed using a confusion matrix on a network traffic dataset. Specifically, 288 out of 289 instances are correctly classified as DDoS slowloris, 245 out of 269 instances as DDoS, and 245 out of 256 instances as Benign. However, there are some misclassifications where the algorithm erroneously categorizes 1 instance of DDoS slowloris as DDoS, 24 instances of DDoS as DDoS slowloris and Benign, and 11 instances of Benign as DDoS slowloris and DDoS.

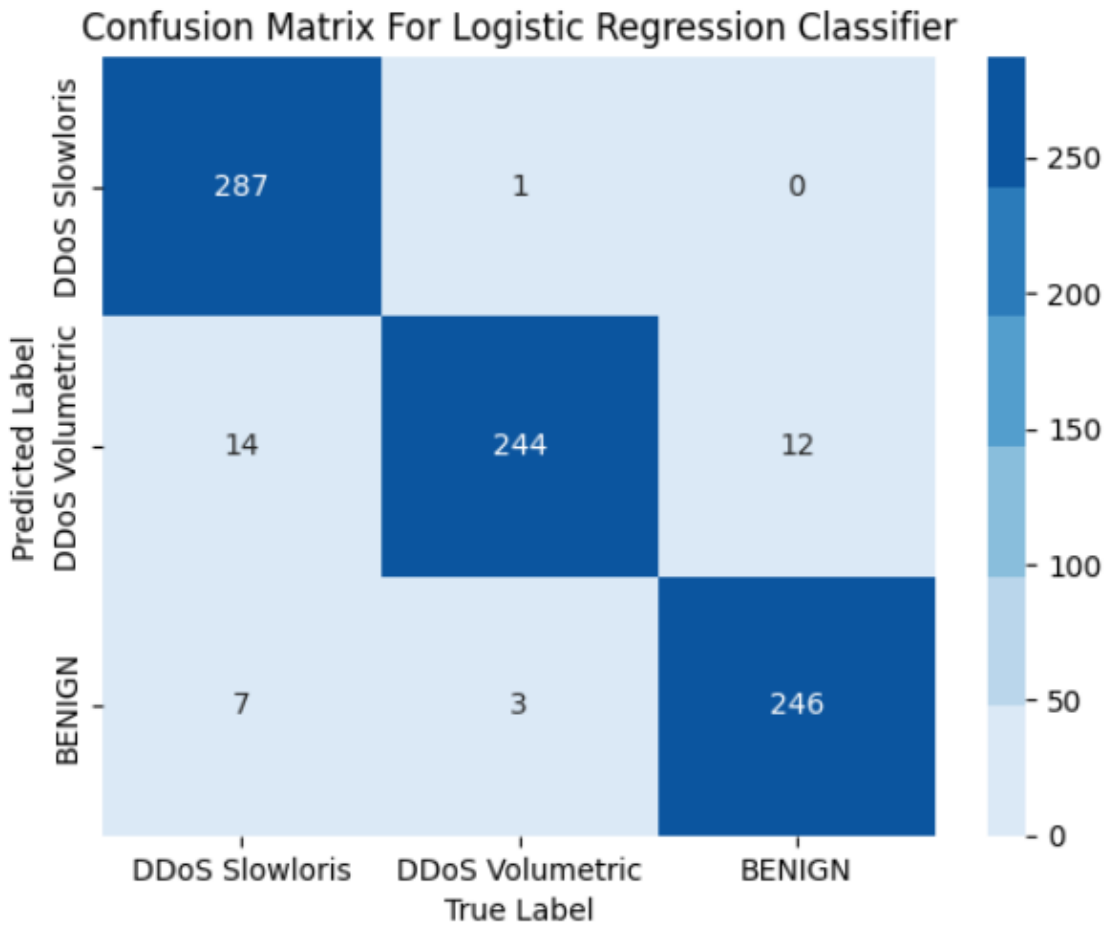

Figure 17: Confusion Matrix for Logistic Regression Classifier

5.3.3 Confusion Matrix for Naïve Bayes Classifier

As illustrated in Figure 18 below, the NB algorithm's performance is evaluated on a network traffic dataset using a confusion matrix. Specifically, 256 out of 289 instances are accurately classified as DDoS slowloris, 249 out of 269 instances as DDoS, and 249 out of 256 instances as Benign. However, there are some misclassifications where the algorithm incorrectly categorizes 33 instances of DDoS slowloris as DDoS, 20 instances of DDoS as DDoS slowloris and Benign, and 7 instances of Benign as DDoS.

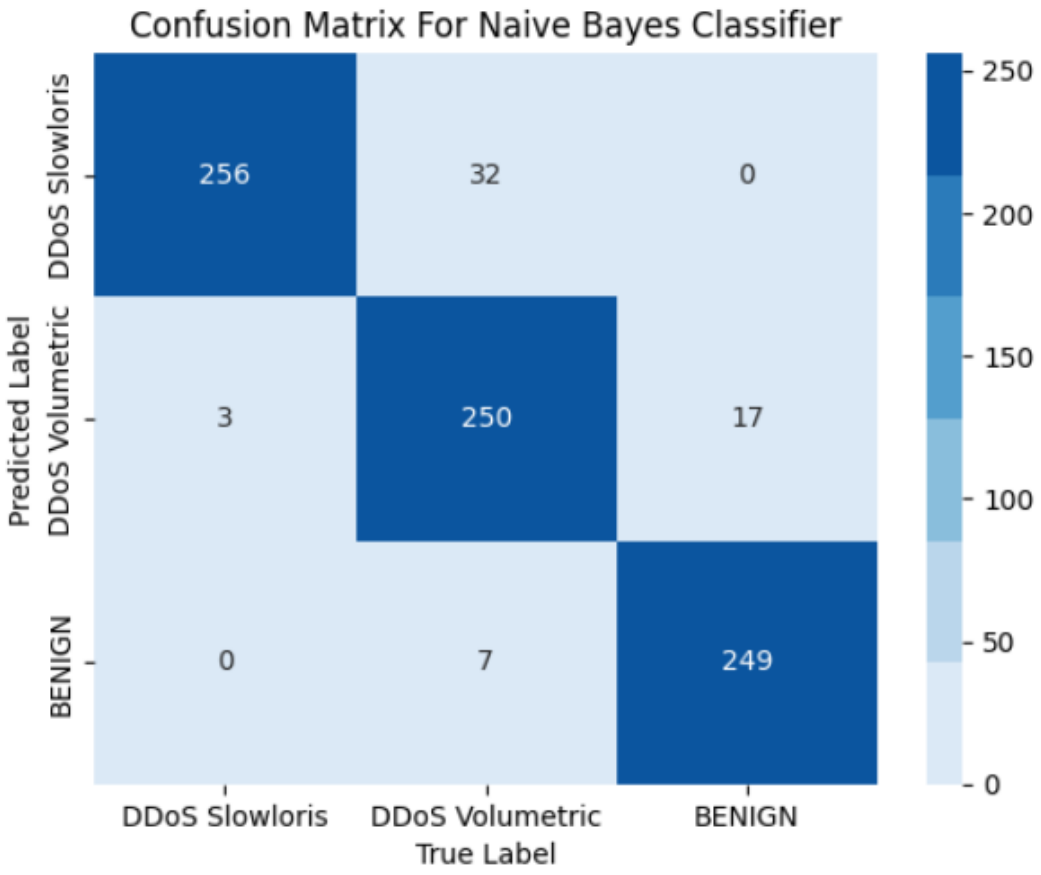

Figure 18: Confusion Matrix for Naïve Bayes Classifier

5.3.4 Confusion Matrix for K-Nearest Neighbor Classifier

As depicted in Figure 19 below, the KNN algorithm's performance is assessed on a network traffic dataset using a confusion matrix. Specifically, 277 out of 289 instances are accurately classified as DDoS slowloris, 260 out of 269 instances as DDoS, and 253 out of 256 instances as Benign. However, there are some misclassifications where the algorithm erroneously categorizes 12 instances of DDoS slowloris as DDoS, 9 instances of DDoS as DDoS slowloris and Benign, and 3 instances of Benign as DDoS.

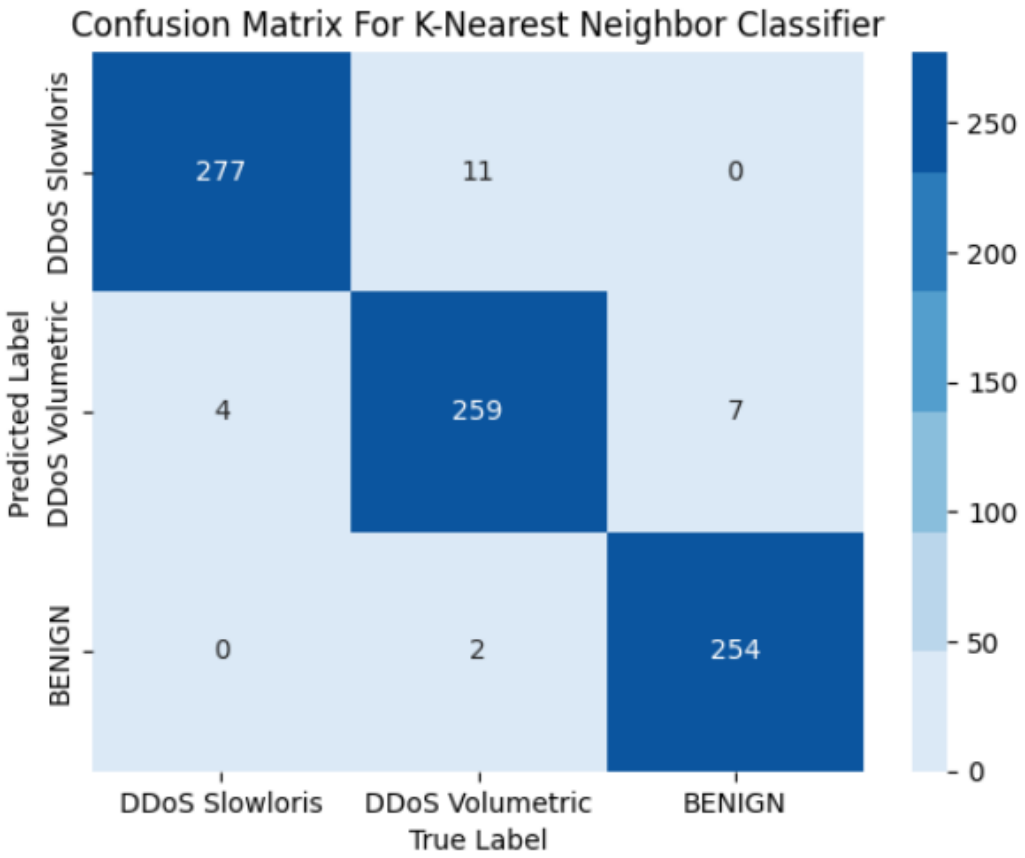

Figure 19: Confusion Matrix for K-Nearest Neighbor Classifier

Models 10-fold CV score of performance Evaluation metrics

Our performance evaluation metrics i.e. precision quantifies the accuracy of positive attack predictions, with a score of 0.96 indicating that 96% of detected attacks were genuine DDoS attacks. This underscores the system's capacity to minimize false alarms, crucial for network security. Recall, or sensitivity, at 0.96 reveals that our system successfully identified and mitigated 96.40% of all actual DDoS attacks, demonstrating its effectiveness in detecting malicious traffic. The F-score, an amalgamation of precision and recall, stands at 0.96, symbolizing a well-balanced trade-off between accuracy and comprehensiveness in attack detection and mitigation, ultimately bolstering network security and stability.

Table 6.1 shown above indicates the 10-fold cross-validation Precision, Recall, F1-score and accuracy of average fold for DT, RF, NB, KNN, LR and SVM models in which SVM scores the highest accuracy of 97.3 than DT, RF, NB, KNN and LR with an accuracy of 96.5, 96.8, 92.3,

95.0, and 96.3 respectively. SVM models outperform the other models in all selected matrices in this study.

Moreover, the performance metrics of the six algorithms are shown in the table below

Table 4: performance metrics result for six models

Models	Precision (%)	Recall (%)	F1-score (%)	10-fold Accuracy result (%)
DT	93.26	97.00	94.69	96.5
RF	93.20	97.00	94.60	96.6
NB	93.30	96.74	94.65	92.3
KNN	93.30	96.74	94.65	95.0
LR	94.44	94.86	94.65	95.6
SVM	96.80	97.44	97.10	97.3

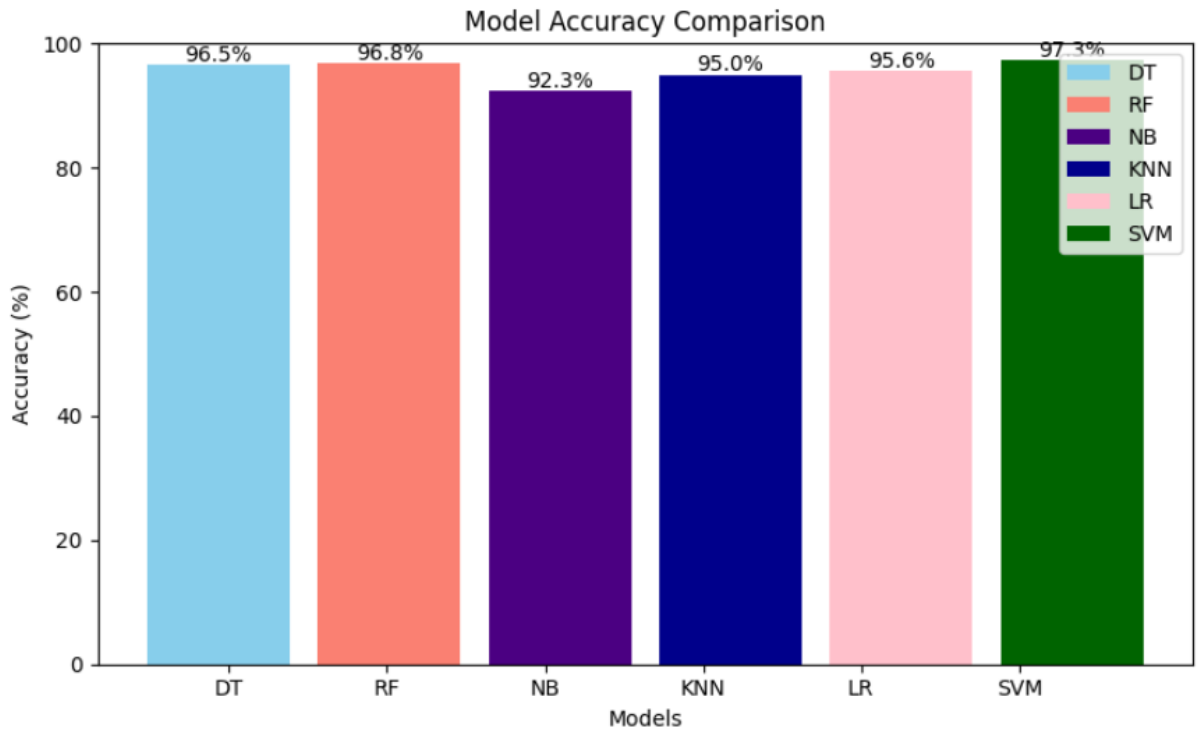

Figure 20: Model accuracy comparison

In the realm of cybersecurity, a learning curve for DDoS (Distributed Denial of Service) detection and mitigation serves as a valuable tool for evaluating the effectiveness of our defense mechanisms. DDoS attacks present a significant threat by inundating online services and networks with massive traffic volumes, making the ability to detect and mitigate them crucial.

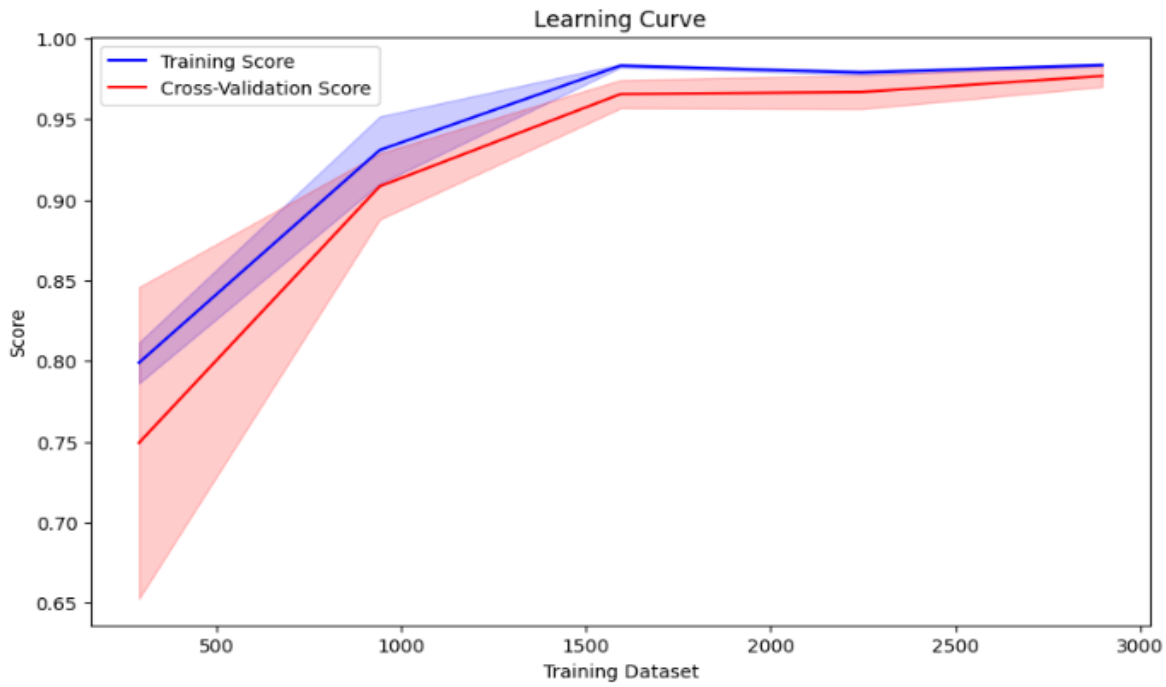

Figure 21: Learning curve for Support Vector Machine Classifier

This learning curve unfolds with the x-axis representing the size of our training dataset used for refining our DDoS detection and mitigation model. Starting on the left side of the curve, where the training dataset is relatively small, our system's performance may be suboptimal. It may struggle to distinguish between legitimate and malicious traffic, resulting in a higher false positive rate. However, as we progress along the curve, incrementally increasing the training data size and potentially fine-tuning our algorithms, our system's performance significantly improves.

This improvement stems from the model's capacity to learn from a more diverse range of attack scenarios and normal traffic patterns, leading to a reduction in the false positive rate and an enhancement in the accuracy of attack detection. The learning curve thus encapsulates the journey of our DDoS detection and mitigation system as it evolves towards greater robustness and reliability.

5.4 Hyperparameter Tuning Result

While automatic hyperparameter tuning is computationally demanding and demands substantial memory resources, this research employs Grid Search hyperparameter tuning with cross-validation, as described in the previous chapter. This choice is made in favor of manual hyperparameter tuning, which is likely to be an inefficient optimization strategy and can lead to subpar and labor-intensive performance. In this study, both grid search and cross-validation random search were tested, and the findings indicate that cross-validation grid search outperforms random search.

Table 5: Optimal Hyperparameters result based on Grid search for SVM

Parameters	**C**	**kernel**	**gamma**
Value	450	rbf	scale

5.5 Discussion

At the beginning of this study, two research questions were encompassed. Hence, this section tries to answer those questions.

Q1. What are the key features that are determinant in the prediction of DDoS attack in Machine Learning models?

To answer this question, Bidirectional feature selection with cross-validation is used to select the determinant attributes. As the finding indicated that the following attributes are the major to the prediction of DDoS attack. Flow_Packets_Sec, min_seg_size_forward, Flow_Bytes_Sec, Fwd_Packet_Length_Max, Fwd_Packet_Length_Std, Max_Packet_Length, Fwd_IAT_Max, Packet_Length_Std, Packet_Length_Variance, Init_Win_bytes_backward, Fwd_IAT_Std, Fwd_IAT_Max and Fwd_IAT_Mean are the most determinant attributes.

Q2. Which classification algorithms will best detect DDoS attacks?

Support vector machine classification algorithm is outperforming other classification algorithms. In addition, to answer this question the researcher selected five classification algorithms and features selection methods and compare the performance of the models with help of 10-fold cross-validation. Six selected models such as SVM, RF, KNN, NB, DT and RF are compared with each other, and then SVM beat others. According to the experiment the new proposed model predicted the DDoS attack with 97.3% accuracy as the best classifier model than other five models.

Generally, the limitation on the reviewed articles on the previous chapter can be stated as following.

- Dynamic nature of attack changes overtime to mimic a legitimate user and those behavior of the attack are hard to capture for both rule-based and signature-based method of detections.
- Previous researches tend to shift their focus to high volumetric DDoS attacks instead of following an inclusive approach.
- Existing research focuses on offline detection and mitigation, however, real-time detection and immediate response to attacks are essential.

This research resolved those limitation by;

1. Employing machine learning models to capture the dynamic nature of DDoS attack instead of traditional rule-based and signature-based detections methods.
2. An inclusive approach of capturing both volumetric i.e., DDoS Volumetric and low-rate i.e., DDoS Slowloris were implied in this research to capture all rounded nature of the attacks instead of being selective.
3. Integrating Apache Kafka method has enabled us to capture real-time flow of packets in the network, which ingested the capability of our machine learning model to capture and process the traffic on the fly.

   Instead of storing the packets for letter analysis that leads to storage complexity on the previous works this research has come up with the way to capture and analysis the traffic without storing them while they are occurring.

CHAPTER SIX

6. CONCLUSION AND FUTURE WORK

Conclusion

In conclusion, this research titled "Real-Time Detection and Mitigation of DDoS Attacks using Machine Learning Approach " represents a significant contribution to the field of cybersecurity and DDoS attack mitigation. Throughout the study, a comprehensive approach was taken to address the growing threat landscape posed by DDoS attacks, with a focus on real-time detection and mitigation using machine learning techniques.

One of the notable accomplishments of this research was the acquisition and preprocessing of a real-world dataset from CICIDS 2019 Dataset sources available on Kaggle. Employing advanced machine learning preprocessing techniques, including dimensionality reduction, handling missing values, and ensuring data balance, ensured the dataset's quality and readiness for model development.

To construct an effective DDoS detection model, feature selection was performed using the Bidirectional feature selection algorithm, resulting in the identification of the most crucial features while discarding less significant ones. This step played a pivotal role in enhancing the model's accuracy and efficiency.

For the better outcome of model training and evaluation, the research skillfully adopted a good methodology, highlighting the commitment to scientific truth. The K-fold cross-validation algorithm was exactly raised, thus filling the research with a shield of robustness, thereby preventing the dangers of overfitting that often bother less sensibly designed models. In this thesis work, six distinct machine learning models, including the Decision Tree, Random Forest, Logistic Regression, K-Nearest Neighbors, Naive Bayes, and Support Vector Machine (SVM), modelled and compared. The proposed model evaluated with performance evaluation metrics, that included the metrics of Precision, F1-Score, Recall, Confusion Metrics, and Accuracy. In this regard, the Support Vector Machine (SVM) model outperforms others with prediction accuracy of 97.3%.

This research also developed mitigation mechanism involving implementation of IP blocking as an effective and practical mitigation strategy.

Generally, this thesis represents a substantial contribution to the field of DDoS attack detection and mitigation. The complete approach, spanning from data attainment and preprocessing to model development and mitigation strategy implementation, addresses the pressing challenges posed by DDoS attacks. The findings affirm the effectiveness of SVM in real-time detection, while the practical application of IP blocking as a mitigation strategy underscores its practicality in countering DDoS attacks. These results collectively pave the way for enhanced cybersecurity practices and the continued pursuit of innovative solutions in the ever-evolving landscape of cyber threats.

Recommendation and Future Works

This study has shown the application of Machine learning for the prediction of DDoS attack to achieve the main objectives. Based on the results of the research the following points are suggested as future works for researchers.

Firstly, it is advisable to continue refining the machine learning models, even though the Support Vector Machine (SVM) demonstrated exceptional performance. Exploring ensemble methods or deep learning architectures could potentially lead to further improvements in the accuracy and robustness of DDoS attack detection.

Secondly, considering the practical deployment of real-time detection and mitigation system in network environments is vital. Assess its real-world applicability, scalability, and efficiency under high-traffic conditions to ensure seamless integration into operational cybersecurity frameworks.

Thirdly, IP blocking can be effective against known attackers or attackers coming from specific IP ranges. However, it can also have limitations, such as being less effective against attacks that use a large number of compromised machines with constantly changing IP addresses. Exploring other techniques of DDoS mitigation is an open area that need to be assessed.

****

Special Acknowledgment This research was funded by Adama science and Technology University

with grant Number:

ASTU/SM-R/847/23

Reference

Ajdani, M., & Ghaffary, H. (2021). Design network intrusion detection system using support vector machine. *International Journal of Communication Systems*, *34*(3). https://doi.org/10.1002/dac.4689

Aldoseri, A., Al-Khalifa, K. N., & Hamouda, A. M. (2023). Re-Thinking Data Strategy and Integration for Artificial Intelligence: Concepts, Opportunities, and Challenges. *Applied Sciences*, *13*(12), 7082. https://doi.org/10.3390/app13127082

Al Lail, M., Garcia, A., & Olivo, S. (2023). Machine Learning for Network Intrusion Detection—A Comparative Study. *Future Internet*, *15*(7). https://doi.org/10.3390/fi15070243

An Overview of Computer Security. (n.d.).

Bajaj, G. L., Syamala Devi, M., Guleria, A., Rai, K., & Syamala Devi Professor, M. (n.d.). *Decision Tree Based Algorithm for Intrusion Detection Multiagent System for Management and Evaluation of Computer Science Examinations View project Multiagent Integrated scheme for Intrusion Detection View project Kajal Rai Saraswat Decision Tree Based Algorithm for Intrusion Detection*. https://www.researchgate.net/publication/298175900

Bazzan, A. L. C. (n.d.). *Traffic as a Complex System: Four Challenges for Computer Science and Engineering Autonomous Control of Traffic Networks View project Redes Interdisciplinares e Multidisciplinares de Pesquisa (RIMPs) View project Traffic as a Complex System: Four Challenges for Computer Science and Engineering*. https://www.researchgate.net/publication/228592374

Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. *Proceedings Second International Workshop on Software and Performance WOSP 2000*, 195–203. https://doi.org/10.1145/350391.350432

Bourke, J., & Wessely, S. (2008). Competent novice: Confidentiality. In *BMJ* (Vol. 336, Issue 7649, pp. 888–891). https://doi.org/10.1136/bmj.39521.357731.BE

C A Padmanabha Reddy, Y., Viswanath, P., & Eswara Reddy, B. (2018). Semi-supervised learning: a brief review. *International Journal of Engineering & Technology*, *7*(1.8), 81. https://doi.org/10.14419/ijet.v7i1.8.9977

Chapter 4 Logistic Regression as a Classifier. (n.d.).

Chen, Z., Yeo, C. K., Lee, B. S., & Lau, C. T. (2018). Power spectrum entropy based detection and mitigation of low-rate DoS attacks. *Computer Networks*, *136*, 80–94. https://doi.org/10.1016/j.comnet.2018.02.029

Christopher U, E., Oluyemi E, A., Oluwasegun I, A., & Ageebee S, F. (2019). Media Access Control (MAC) Protocols: An Overview. *International Journal of Advances in Scientific Research and Engineering*, *5*(8), 142–149. https://doi.org/10.31695/ijasre.2019.33462

Classification: True vs. False and Positive vs. Negative | Machine Learning | Google for Developers. (n.d.). Retrieved September 9, 2023, from https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

Daniel, J., & Martin, J. H. (2023). *Speech and Language Processing*.

Dayanandam, G., Rao, T. V., Bujji Babu, D., & Nalini Durga, S. (2019). DDoS attacks—Analysis and prevention. In *Lecture Notes in Networks and Systems* (Vol. 32, pp. 1–10). Springer. https://doi.org/10.1007/978-981-10-8201-6_1

Farik, M., Lal, N. A., & Prasad, S. (2016). A Review Of Authentication Methods. *Article in International Journal of Scientific & Technology Research*, *5*(11). www.ijstr.org

Farnaaz, N., & Jabbar, M. A. (2016). Random Forest Modeling for Network Intrusion Detection System. *Procedia Computer Science*, *89*, 213–217. https://doi.org/10.1016/j.procs.2016.06.047

Farnham, B., Tokyo, S., Boston, B., Sebastopol, F., & Beijing, T. (n.d.). *Distributed Denial of Service (DDoS) Practical Detection and Defense*.

Gaurav, A., Zhou, Z., Tai Chui, K., Colace, F., Chaurasia, P., & Hsu, C.-H. (2021). *A Novel Approach for DDoS Attack Detection Using Big Data and Machine Learning*. http://ceur-ws.org

Gu, Q., & Liu, P. (n.d.). *Denial of Service Attacks*.

Haripriya, L., & Jabbar, M. A. (2018). Role of Machine Learning in Intrusion Detection System: Review. *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, 925–929. https://doi.org/10.1109/ICECA.2018.8474576

IEEE Instrumentation & Measurement Magazine Network Measurement: The Context. (2008).

Jonsson, M., Institute of Electrical and Electronics Engineers. Sweden Section. VT/COM/IT Chapter, IFIP Working Group 6.10, Sankt-Peterburgskiĭ nat͡sional′nyĭ issledovatel′skiĭ universitet informat͡sionnykh tekhnologiĭ, mekhaniki i optiki, & Institute of Electrical and Electronics Engineers. (n.d.). *Proceedings of 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM) : September 13-15, 2016, Halmstad, Sweden*.

Karhunen, J., Raiko, T., & Cho, K. (n.d.). *Unsupervised Deep Learning: A Short Review*.

Karimi, Z. (n.d.). *Confusion Matrix Some of the authors of this publication are also working on these related projects: Data Cleaning Process View project*. https://www.researchgate.net/publication/355096788

Kausar, N., Belhaouari Samir, B., Abdullah, A., Ahmad, I., & Hussain, M. (2011). A Review of Classification Approaches Using Support Vector Machine in Intrusion Detection. In *CCIS* (Vol. 253).

Kumari, K., & Mrunalini, M. (2022). Detecting Denial of Service attacks using machine learning algorithms. *Journal of Big Data*, *9*(1). https://doi.org/10.1186/s40537-022-00616-0

Kumari, N., & Malhotra, V. (2022). Intrusion Detection System Using Machine Learning: An Overview. *International Research Journal of Engineering and Technology*. https://doi.org/10.13140/RG.2.2.20109.41441

Kumar, V. (2014). Feature Selection: A literature Review. *The Smart Computing Review*, *4*(3). https://doi.org/10.6029/smartcr.2014.03.007

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. In *ACM Computing Surveys* (Vol. 50, Issue 6). Association for Computing Machinery. https://doi.org/10.1145/3136625

Lima Filho, F. S. De, Silveira, F. A. F., De Medeiros Brito Junior, A., Vargas-Solar, G., & Silveira, L. F. (2019). Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning. *Security and Communication Networks*, *2019*. https://doi.org/10.1155/2019/1574749

Mahajan, D., & Sachdeva, M. (2013). DDoS Attack Prevention and Mitigation Techniques - A Review. *International Journal of Computer Applications*, *67*(19), 21–24. https://doi.org/10.5120/11504-7221

Mahjabin, T., Xiao, Y., Sun, G., & Jiang, W. (2017). A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, *13*(12). https://doi.org/10.1177/1550147717741463

Majeed Alhammadi, N. A., Hameed Zaboon, K., & Abdulhadi Abdullah, A. (2022). A Review of the Common DDoS Attack: Types and Protection Approaches Based on Artificial Intelligence. *Fusion: Practice and Applications*, *7*(1), 8–14. https://doi.org/10.54216/FPA.070101

Martin Ward Powers, D. (2011). *Applications of Language and Learning Technology View project Evolutionary Optimization of Brain Computer Interface: Doing More with Less View project EVALUATION: FROM PRECISION, RECALL AND F-MEASURE TO ROC, INFORMEDNESS, MARKEDNESS & CORRELATION*. *2*(1), 37–63. https://doi.org/10.9735/2229-3981

M, H., & M.N, S. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, *5*(2), 01–11. https://doi.org/10.5121/ijdkp.2015.5201

Michaud, E. J., Liu, Z., & Tegmark, M. (2022). *Precision Machine Learning*. https://doi.org/10.3390/e25010175

Mihoub, A., Fredj, O. Ben, Cheikhrouhou, O., Derhab, A., & Krichen, M. (2022). Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques. *Computers and Electrical Engineering*, *98*. https://doi.org/10.1016/j.compeleceng.2022.107716

Najafimehr, M., Zarifzadeh, S., & Mostafavi, S. (2023). DDoS attacks and machine-learning-based detection methods: A survey and taxonomy. *Engineering Reports*, e12697. https://doi.org/10.1002/ENG2.12697

Oladipupo, T. (2010). Machine Learning Overview. In *New Advances in Machine Learning*. InTech. https://doi.org/10.5772/9374

Prriyadarshini, M. A., & Renuka Devi, S. (2021). Detection of DDoS attacks using supervised learning technique. *Journal of Physics: Conference Series*, *1716*(1). https://doi.org/10.1088/1742-6596/1716/1/012057

Rhohim, A., Suryani, V., & Nugroho, M. A. (n.d.). *Denial of Service Traffic Validation Using K-Fold Cross Validation on Software Defined Network*.

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. In *SN Computer Science* (Vol. 2, Issue 3). Springer. https://doi.org/10.1007/s42979-021-00592-x

Shrivas, A. K., & Mishra, K. (n.d.). *Intrusion Detection System for Classification of Attacks with Cross Validation*. www.ijesi.org

Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006). Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation. *AAAI Workshop - Technical Report*, *WS-06-06*, 24–29. https://doi.org/10.1007/11941439_114

Srinivas, B., & Swapna, S. (n.d.). *A Comprehensive Overview on Types of Machine Learning*. www.rrjournals.com

Srivastava, M. S., Joshi, N., & Gaur, M. (2013). A Review Paper on Feature Selection Methodologies and Their Applications. In *International Journal of Engineering Research* (Vol. 7, Issue 6). www.ijerd.com

Sweeney, C., Ennis, E., Mulvenna, M., Bond, R., & O'neill, S. (2022). How Machine Learning Classification Accuracy Changes in a Happiness Dataset with Different Demographic Groups. *Computers*, *11*(5). https://doi.org/10.3390/computers11050083

Torabi, M., Udzir, I., Abdullah, M. T., & Yaakob, R. (n.d.). A Review on Feature Selection and Ensemble Techniques for Intrusion Detection System. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 12, Issue 5). www.ijacsa.thesai.org

True Positive Rate Definition | Encord. (n.d.). Retrieved September 9, 2023, from https://encord.com/glossary/true-positive-rate-definition/

Vijaya, R., Reddy, K., & Ravi Babu, U. (n.d.). *A Review on Classification Techniques in Machine Learning*.

Wang, S., Tang, J., & Liu, H. (2016). Feature Selection. In *Encyclopedia of Machine Learning and Data Mining* (pp. 1–9). Springer US. https://doi.org/10.1007/978-1-4899-7502-7_101-1

What is False Positive Rate. (n.d.). Retrieved September 9, 2023, from https://www.iguazio.com/glossary/false-positive-rate/

Zulkii Hasan, M., Zunnurain Hussain, M., Zeeshan Nazar, M., Anjum, F., Zulkifl Hasan, M., Mustafa, G., Atif Yaqub, M., Mustafa, M., Hussain Chuhan, S., Mubarak, Z., Ahmad Siddiqui, A., Moiz Qureshi, A., Author, C., & Zulkifl Hasan, M. (2023). *Detection & Mitigation of DDOS Attack*. https://doi.org/10.21203/rs.3.rs-2586448/v1

APPENDIX

Appendix A: Dataset Description

Flow_Packets_Sec: This feature represents the rate at which packets are transmitted within a network flow.

min_seg_size_forward: This feature indicates the minimum segment size observed in the forward direction of network traffic.

Flow_Bytes_Sec: Similar to Flow_Packets_Sec, this feature measures the rate at which bytes are transmitted within a network flow.

Fwd_Packet_Length_Max: This feature represents the maximum length of packets in the forward direction of traffic.

Fwd_Packet_Length_Std: This is the standard deviation of packet lengths in the forward direction.

Max_Packet_Length: This attribute captures the maximum length of any packet in the dataset, regardless of direction.

Fwd_IAT_Max: Forward Inter-Arrival Time (IAT) represents the time between consecutive packets in the forward direction.

Fwd_IAT_Max: is the maximum inter-arrival time observed in the forward direction.

Packet_Length_Std: This is the standard deviation of packet lengths across all directions (both forward and backward).

Packet_Length_Variance: This feature provides the variance in packet lengths, indicating how spread out the packet lengths are.

Init_Win_bytes_backward: This feature pertains to the initial window size in bytes for the backward direction of the flow.

Fwd_IAT_Std: Similar to Fwd_IAT_Max, this attribute deals with inter-arrival times in the forward direction. Fwd_IAT_Std represents the standard deviation of these times, highlighting variations in packet arrival patterns.

Fwd_IAT_Max: This is the maximum forward inter-arrival time, which measures the longest delay between consecutive packets in the forward direction.

Fwd_IAT_Mean: This feature calculates the mean (average) forward inter-arrival time, providing insight into the typical packet arrival rate in the forward direction.

Appendix A1: Samples dataset

1	Source.IP	Source.Port	Destination.I	Destination.Pi	Total_Length	Fwd_Packet_	Fwd_Pack	Fwd_Pack	Fw	Active_Me	Active_Sto	Active_Ma	Active_Mi	Idle_Mear	Idle_Std	Idle_Max	Idle_Min	Label
2	10.200.7.4	3128	192.168.180.5	57740	0	189	0	50	8(	1459689	1243131	2342653	38077		5917398	24671501	10587946	DDoS
3	172.19.1.46	52422	10.200.7.7	3128	0	353	0	48.45	1:	739228.5	743103.5	1264682	213775	49700000	41400000	79000000	20500000	DDoS
4	10.200.7.7	3128	172.19.1.46	52422	0	0	0	0		0	0	0	0	0	0	0	0	BENIC
5	50.31.185.39	80	10.200.7.217	38848	244	37	37	37		0	0	0	0	0	0	0	0	BENIC
6	50.31.185.39	80	10.200.7.217	38848	389	423	0	143	2(	0	0	0	0	0	0	0	0	BENIC
7	192.168.72.4	55961	10.200.7.7	3128	15703	1525	0	419.1667	6(	0	0	0	0	0	0	0	0	BENIC
8	10.200.7.6	3128	172.19.1.56	50004	0	168	0	33.16667			0	477294	477294		0	42076895	42076895	DDoS
9	192.168.72.4	55963	10.200.7.7	3128	177	49	49	49		0	0	0	0	0	0	0	0	BENIC
10	192.168.10.4	51848	10.200.7.6	3128	0	374	0	68	1:	7008534	0	7008534	7008534	8013363	0	8013363	8013363	DDoS
11	68.67.178.19	443	10.200.7.217	57300	0	314	0	24.15385	8:	0	0	0	0		0	38200000		DDoS
12	192.168.72.4	55977	10.200.7.7	3128	0	357	0	68.93333	1(	2168000	0	2168000	2168000	37400000	0	37400000		DDoS
13	10.200.7.4	3128	192.168.180.5	57740	0	189	0	50	8(	1459689	1243131	2342653	38077		5917398	24671501	10587946	DDoS
14	10.200.7.4	3128	192.168.180.5	57740	0	0	0	0								0	0	DDoS
15	10.200.7.4	3128	192.168.180.5	57740	158	32	32	32		0	0	0	0	0	0	0	0	BENIC
16	10.200.7.4	3128	192.168.180.5	57740	264	30	30	30		0	0	0	0	0	0	0	0	BENIC
17	10.200.7.6	3128	172.19.1.45	50227	0	338	0	67.6	1:	0	0	0	0	0	0	0	0	DDoS
18	212.124.124	443	10.200.7.194	44447	0	0	0	0		0	0	0	0	0	0	0	0	BENIC
19	10.200.7.4	3128	192.168.180.5	57741	0	355	0	52.07692	1:		0	494845	494845	61900000	0		61900000	DDoS
20	10.200.7.4	3128	192.168.180.5	57741	0	0	0	0			0	0	0	0	0	0	0	BENIC
21	10.200.7.4	3128	192.168.180.5	57741	30	316	0	75.33333	1:	0	0	0	0	0	0	0	0	BENIC
22	172.217.30.1	443	10.200.7.217	41526	0	322	0	24.76923	8:		0	29188	29188		0		40600000	DDoS
23	172.217.30.1	443	10.200.7.217	41526	0	0	0	0		0	0	0	0	0	0	0	0	BENIC
24	10.200.7.9	3128	192.168.180.3	1978	0	0	0	0		0	0	0	0	0	0	0	0	DDoS
25	10.200.7.9	3128	192.168.180.3	1978	196	27	27	27		0	0	0	0	0	0	0	0	BENIC
26	10.200.7.9	3128	192.168.180.3	1978	167	85	85	85		0	0	0	0	0	0	0	0	BENIC
27	192.168.32.4	54500	10.200.7.9	3128	242	31	31	31		0	0	0	0	0	0	0	0	BENIC
28	10.200.7.9	3128	192.168.32.49	54500	0	344	0	57.33333	1(	0	0	0	0	0	0	0	0	DDoS

Appendix B: sample code

B1: SVM sample code

```
# Define the Bidirectional Feature Selection (BFS)
bfs_selector = SelectKBest(score_func=f_classif, k='all')
X_train_selected = bfs_selector.fit_transform(X_train, y_train)
X_test_selected = bfs_selector.transform(X_test)

# Define the SVM hyperparameter grid
svm_param_grid = {
    'C': [0.1, 450, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto', 0.1]
}

# Create an SVM classifier
svm_classifier = SVC()
svm_grid_search = GridSearchCV(svm_classifier, svm_param_grid,
cv=KFold(n_splits=10, shuffle=True, random_state=42), verbose=2, n_jobs=-1)
svm_grid_search.fit(X_train_selected, y_train)

# Get the best SVM model
best_svm_classifier = svm_grid_search.best_estimator_

# Evaluate the SVM model
y_pred = best_svm_classifier.predict(X_test_selected)
print('SVM Classifier Accuracy:', metrics.accuracy_score(y_test, y_pred))
confusion = confusion_matrix(y_test, y_pred)
print(confusion)
print("SVM Classifier Accuracy:", svmClassifier.score(X_test, y_test))
print(classification_report(y_test, y_pred))

precision, recall, f1_score, support = precision_recall_fscore_support
 (y_test, y_pred, average='macro')
print("Macro Precision:", precision)
print("Macro Recall:", recall)
print("Macro F1 Score:", f1_score)

# Define the k-NN hyperparameter grid
knn_param_grid = {
```

B2: Sample Code for performing Data balancing and drawing Confusion metrics.

```
# Define your features and labels
X = data[features]
y = data["Label"]

# Create a SMOTE instance
smote = SMOTE(random_state=42)

# Resample the dataset
X_resampled, y_resampled = smote.fit_resample(X, y)
```

```
y_pred = logRegression.predict(X_test)
logRegressionccuracy = metrics.accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
print(confusion)
sns.heatmap(confusion, annot=True, fmt="d")
plt.xlabel("True Label")
plt.ylabel("Predicted Label")
plt.title("Confusion Matrix For Logistic Regression Classiffier")
plt.show()
```

Appendix C: Accuracy result for each fold in SVM model

```
Fold 1: Accuracy = 96.99%
----------------------------------------
Fold 2: Accuracy = 97.74%
----------------------------------------
Fold 3: Accuracy = 96.87%
----------------------------------------
Fold 4: Accuracy = 97.29%
----------------------------------------
Fold 5: Accuracy = 97.59%
----------------------------------------
Fold 6: Accuracy = 97.55%
----------------------------------------
Fold 7: Accuracy = 97.10%
----------------------------------------
Fold 8: Accuracy = 97.82%
----------------------------------------
Fold 9: Accuracy = 97.26%
----------------------------------------
Fold 10: Accuracy = 97.05%
----------------------------------------
========================================
Average Accuracy: 97.33%
```