Machine Learning Based Approach for Detecting Distributed Denial of Service Attack

by

Qozeem Adeniyi Adeshina

A project report submitted in conformity with the requirements
for the degree of Master of Science in Information Technology

Department of Mathematical and Computer Sciences
Faculty of Graduate Studies
Concordia University of Edmonton

Abstract

All elements of the IT industry are expanding, including bandwidth, storage, processing speed. As a result, there are now more cyber threats and attacks, necessitating a creative and predictive security approach that employs cutting-edge technology to combat the danger. The trends will be monitored, and adequate analysis from various sets of data will be utilized to build a model that is based on the information available. Distributed Denial of Service (DDoS) is one of the most prevalent dangers and attacks wreaking havoc on internet-connected computer equipment. This study compares the performance of several machine learning-based classifiers for detecting DDoS assaults before they occur. This experiment made use of data from the benchmark KDD-Cup-1999 DDoS attack. To choose essential characteristics in the context of DDoS detection, I created three distinct types of feature selection techniques. The findings revealed that feature selection approaches can assist domain specialists in understanding the intrusion system's hidden key patterns and features during DDoS detection. DNN-based deep learning and semi-supervised learning method were also compared with the ML-based classifiers output. The suggested model learns to recognize regular network traffic to detect ICMP, TCP, and UDP DDoS traffic as it arrives. Experiments show that machine learning algorithms may correctly classify the traffic into regular and DDoS. This discovery has long-term implications in a variety of sectors, including national defence, financial institutions, healthcare, and other businesses where sophisticated intrusion detection techniques are required. In the future, I would want to apply similar approaches to a variety of datasets.

This project is dedicated to the loving memory of my parents, Mudashir and Aminat Adeshina. May Allah (SWT) overlook their shortcomings.

Acknowledgments

Firstly, praise and gratitude to Allah (SWT), for showering His blessings on me during my research work, allowing me to successfully complete the study.

I would like to convey my heartfelt thanks to Dr. Baidya Nath Saha, Assistant Professor and Head of the Department of Mathematical and Computer Sciences at Concordia University of Edmonton, for allowing me to do this research study and providing me with invaluable guidance during the process. His energy, vision, honesty, and determination have all left an indelible impression on me. He showed me how to conduct research and present my findings in the most clear and concise manner possible. Working and studying under his tutelage was an honor and privilege. I am appreciative for everything he has done for me.

My wife, Ghaniyat, has my undying gratitude for her love, prayers, care, patience, unwavering support, and sacrifices. My thankfulness to her for the companionship, sensitivity, and wonderful sense of humor. I am also grateful to my daughters, Mueezah and Hameedah Adeshina, for their love and prayers in helping me finish this project.

I also want to thank my siblings, Peju, Kunle, and Bukola with their spouses Lekan, Olaitan and Gbanga respectively, for their moral and financial support, as well as their prayers. Their contributions to this project work is immense. The fatherly and motherly roles they have played are steadfast and cannot be overemphasized.

Mr. Akeem Adeshina and his family deserve a special thank you for their genuine support from the start of my studies till today. The words of encouragement and help from him and his wife have a significant impact on my self-esteem. I owe them a huge debt of gratitude.

My immeasurable thanks go to my friends, who are more like brothers to me - Jide, Bunmi, Razaq and Mayowa (PH) for the huge support and keen interest shown through the successful completion of this thesis. They have always been very encouraging in their advice, feedback and have always made me feel confident in my skills and abilities after speaking with them on the phone.

I would want to extend my thanks to my classmates, the pioneer set of MSc. IT 2020/2021, Concordia University of Edmonton for their collaborative support with which we all benefited from. My final appreciation goes to everyone who has helped me accomplish the research work, whether directly or indirectly.

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Background 5

1.3 Problem Statement 7

1.4 State of the art 8

1.5 Contribution of Thesis 9

1.6 Organization of Thesis 9

2 Literature Review **10**

2.1 DDoS Detection Using ML 10

2.2 DDoS Detection Using DL 12

3 Machine Learning and Deep Learning Based Strategies for DDoS Detection **15**

3.1 Model Architecture 15

3.2 Dataset Generation 16

3.3 Data Pre-processing 17

3.3.1 Data Transformation 17

3.3.2 Data Discretization 18

3.3.3 Data Normalization 18

3.3.4 Domain Knowledge 18

3.4 Feature Selection Techniques 18

3.4.1 Filter-based Methods 19

3.4.2 Wrapper-based methods 19

3.4.3 Embedded-based methods 19

3.5 Classifier Models 20

4 Feature Selection and Visualization **21**

4.1 Feature Selection 21

4.1.1 Filter method . . . 22
4.1.2 Wrapper Method . . . 23
4.1.3 Embedded Method . . . 26
4.1.4 Bayesian Method . . . 28
4.2 Visualization . . . 29
4.2.1 RadViz . . . 29
4.2.2 Parallel Coordinates . . . 30

5 ML Algorithms 31
5.1 Supervised Learning algorithms . . . 31
5.1.1 K-Nearest Neighbor . . . 31
5.1.2 Gradient Boosting . . . 32
5.1.3 AdaBoost . . . 33
5.1.4 Support Vector Machine . . . 33
5.1.5 Naive Bayes . . . 34
5.1.6 Neural Network . . . 35
5.1.7 Decision Tree . . . 36
5.1.8 Random Forest . . . 37
5.1.9 Logistic Regression . . . 37
5.1.10 Multinomial Naïve Bayes . . . 38
5.2 Semi-supervised Learning Algorithms . . . 38
5.3 Deep Learning Algorithms . . . 39
5.4 SMAC . . . 40

6 Experimental Results 41
6.1 Data Description . . . 41
6.2 Exploratory Data Analysis . . . 43
6.3 Feature Selection Visualization . . . 46
6.3.1 Feature Selection Results . . . 46
6.3.2 Visualization Results . . . 48
6.4 Result of ML-based algorithms . . . 51
6.5 Result of Semi-supervised Learning . . . 54
6.6 Result of DL Algorithm . . . 55

7 Conclusion and Future Works 56

Bibliography 57

List of Tables

4.1 Major differences between 3 feature selection techniques 22

6.1 Basic feature names and descriptions 42
6.2 Traffic feature names and descriptions 42
6.3 Content feature names and descriptions 43
6.4 Selected Features of Variance Threshod 46
6.5 Selected Features of Pearson Correlation 47
6.6 Classifier results under Variance Threshold 52
6.7 Classifier results under Mutual Information 52
6.8 Classifier results under ANOVA F-value 52
6.9 Classifier results under Selectk Best 53
6.10 Classifier results under Pearson Correlation 53
6.11 Classifier results under RFE Wrapper Method 53
6.12 Classifier results under Embedded Method 54
6.13 Supervised Classification 54
6.14 Semi-supervised classification 55
6.15 Comparison between supervised and Semi-supervised classification . . 55
6.16 Deep learning results 55

List of Figures

1.1 Architecture of DDoS Attack 2
1.2 DDoS Attack Taxonomy 3
1.3 TCP Flood Attack 4
1.4 UDP Flood Attack 4
1.5 ICMP Flood Attack 5

3.1 Model Architecture for DDoS attack detection 16
3.2 Filter Model 19
3.3 Wrapper Model 20
3.4 Embedded Model 20

4.1 Filter-based method of feature selection 21
4.2 Wrapper-based method of feature selection 24
4.3 Embedded-based method of feature selection 26

5.1 Architecture of Semi-supervised Learning 39

6.1 Bar plot showing frequency distribution of the dataset 43
6.2 Heat map showing feature to feature relationship 44
6.3 Principal Component Analysis 45
6.4 t-Distributed Stochastic Neighbor Embedding 45
6.5 Selected Features of Mutual Information 46
6.6 Selected Features of ANOVA F-Value 47
6.7 Selected Features of SelectK Best 47
6.8 Weights of RFE Wrapper Features 48
6.9 RaViz visualization of mutual information features 49
6.10 RaViz visualization of ANOVA F-value features 49
6.11 RaViz visualization of SelectK Best features 50
6.12 Parallel Coordinate Plot for Mutual Information Features 50
6.13 Parallel Coordinate Plot for ANOVAF Value Features 51

6.14 Parallel Coordinate Plot for Selectk Best Features 51

Chapter 1

Introduction

1.1 Introduction

Cyber threats are a major concern for businesses and organizations across the globe. It is an activity intended by cybercriminals to utilize whatever resources they have to take aggressive action against a single computer device or a network of machines. A cyber-attack might manifest in a variety of ways. One of the commonest of them all is Distributed Denial-of-Service (DDoS), in which a large amount of traffic is directed towards a specific target in order to knock it down. When carrying out their harmful acts, these criminals have a variety of motives. It can include stealing data for personal benefit, financial gain, or a competitive edge, among other things. To combat these rapidly increasing attacks, it is necessary to learn how to anticipate attacks and what security measures to use to safeguard corporate enterprises.

DDoS is very difficult to fight. It is one of the most utilized techniques of attack. The reason for this problem is because the attack appears from several IP address locations throughout the internet at the same time, making it more difficult to pinpoint the source of the attack. DDoS attacks can affect a large number of devices connected to the internet. The attack can be directed at any network device, although it is most used against web servers. The attack is carried out by pumping requests that gradually overwhelm and overwork the target device's resources, such as CPU utilization, bandwidth, memory, and so on, preventing legitimate requests from being completed and so obstructing proper functioning.

A DDoS assault employs a large number of computers to conduct a coordinated DOS attack against one or more targets. By using the resources of numerous unaware participant computers, which act as attack platforms, the perpetrator is able

to greatly increase the efficacy of the DOS.
DDoS has four compositions as described below [1]. The diagrammatic illustration is also shown in Fig 1.1

The attackers – they are the central controller. The entire sequence of events is controlled by the attacker.
The handlers – they are compromised hosts, who are capable of commanding a large number of agents.
The attack agents – they are also called zombie hosts, who are in charge of sending a stream of packets to the target victim.
A victim – also called the target host.

The susceptible targets (Handlers) in the control stage are used to control another set of vulnerable targets (Agents). The attacker is the central controller, who employs handlers and agents to launch a dispersed attack traffic (attack stage) on the target. Eventually, the attack flow will overwhelm the final target. Potential attackers now have access to sophisticated and powerful DDoS toolkits, raising the risk of being a victim of a DOS or DDoS attack. Trinoo, TFN, Stacheldraht, TFNZK, mstream, and Shaft are some of the most well-known DDoS tools. DDoS attacks come in a number

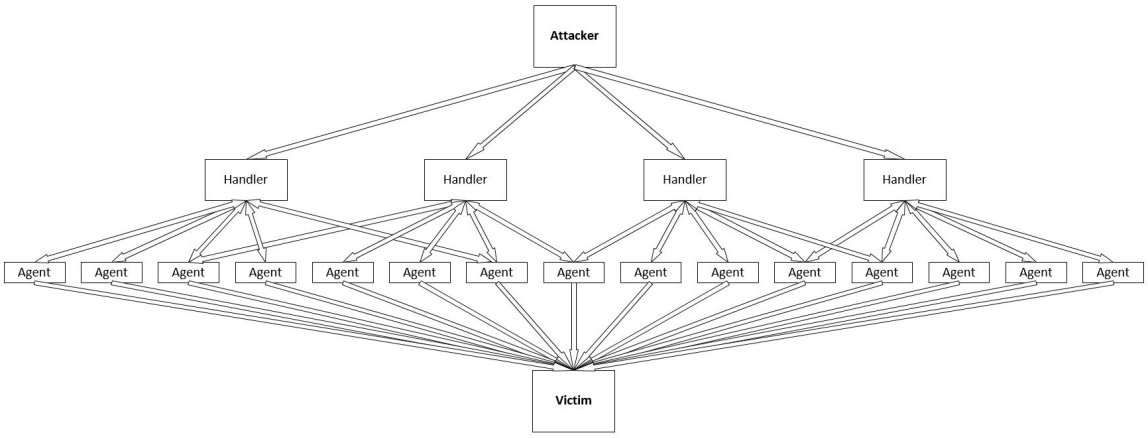

Figure 1.1: Architecture of DDoS Attack

of forms referred to as DDoS Attack Taxonomy. Active and passive DDoS attacks are the two types of DDoS attacks. Even when there is no congestion, packet dropping is a form of passive attack in which a node loses part, or all of the data packets delivered to it for onward forwarding. DDoS attacks are divided into two categories: bandwidth depletion and resource depletion. Figure 1.2 depicts the classification of various DDoS attacks. DDoS attacks can also be categorized into three groups viz
Protocol based attack: This form of DDoS focuses on abusing server resources by sending malicious connection requests measured in packets per second to Layer 3 and

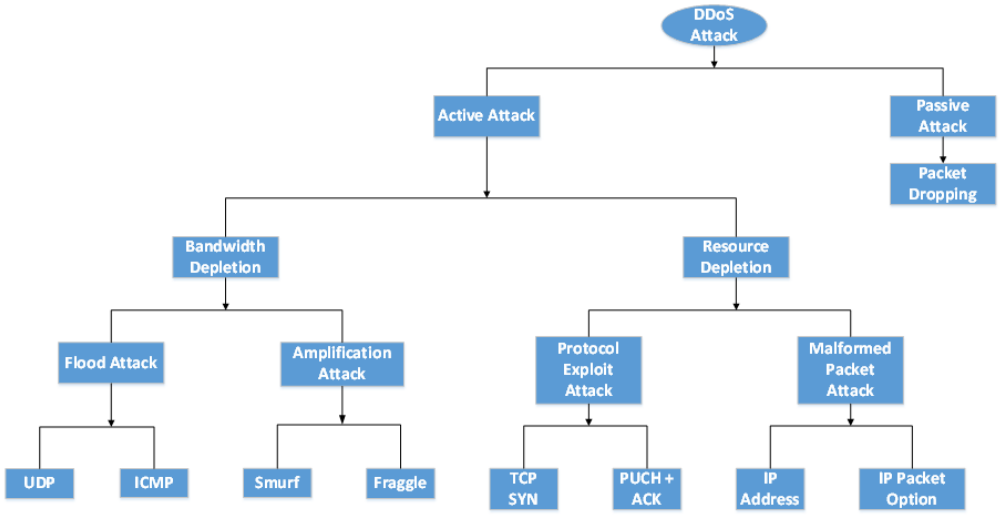

Figure 1.2: DDoS Attack Taxonomy

Layer 4 protocol communications (Pps).

Application based attack: This sort of assault targets online applications and is considered the most sophisticated and dangerous. It takes advantage of flaws in the system and floods requests at layer 7 (Application layer), which is measured in Requests per second (Rps).

Volume based attack: Synchronize (SYN) Flood, User Datagram Protocol (UDP) Flood, Internet Control Message Protocol (ICMP) Flood, and other faked packet floods are used to overload the bandwidth of the target machine or network, which is measured in bits per second (bps). This study focuses on volumetric assaults, such as TCP SYN, UPD, and ICMP DDoS packets.

- **TCP SYN Flood Attack**: This sort of denial-of-service attack occurs when an attacker quickly establishes a connection to a server without completing it. SYN, SYN-ACK, ACK (TCP 3-way handshake) is a procedure that establishes a connection between the initiator and the destination in a TCP/IP network. To create a TCP connection, the initiator sends a SYN to the destination computer, which responds with a SYN-ACK, and the initiator responds with an ACK to complete the circuit.

  As shown in Fig 1.3, the attacker sends repeated SYN packets to every port on the targeted server in a SYN flood attack. Unaware of the assault, the server will have no choice but to respond with an identical amount of SYN-ACK. The attacked server will wait for the ACK of its SYN-ACK packet. It wastes resources

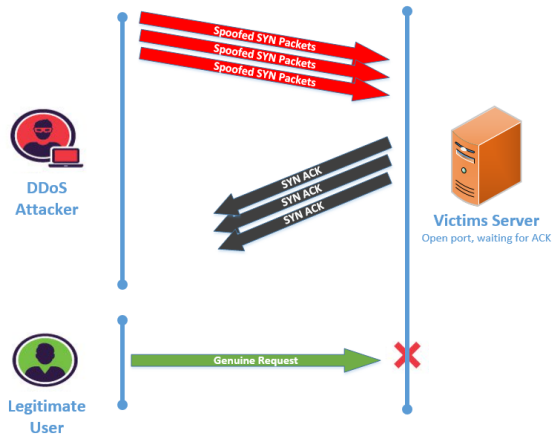

Figure 1.3: TCP Flood Attack

waiting for half-opened connections; before the connection can time out, another SYN packet will come, and the system will have a rising number of half-opened connections, rendering it unresponsive to real traffic.

- **UDP Flood Attack**: Another form of volume-based DDoS assault is a UDP flood, in which the threat actors bombard random ports on the target host with IP packets carrying User Datagram Protocol (UDP) packets.

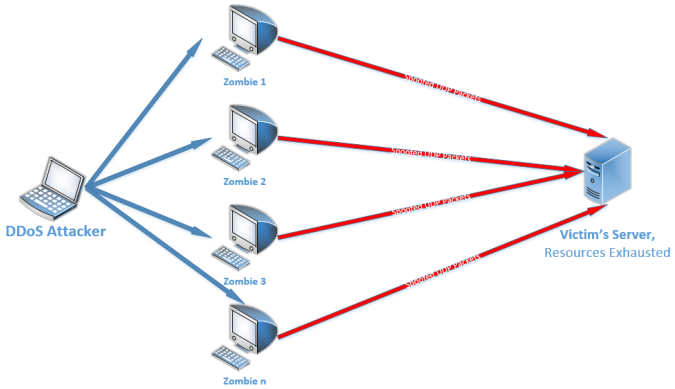

Figure 1.4: UDP Flood Attack

The target of this sort of assault, as illustrated in Fig 1.4, searches for applications connected with these datagrams but is unable to locate them. It then returns to the sender with a "Destination Unreachable" packet. As a result of being inundated with such a torrent, the system becomes saturated and hence unresponsive to valid traffic.

- **ICMP Flood Attack**: This assault, also known as a Ping flood attack, is a

popular Denial-of-Service (DoS) attack in which an attacker sends a large number of ICMP echo-requests to a single device (pings).

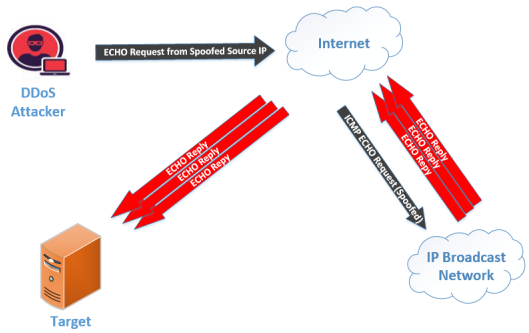

Figure 1.5: ICMP Flood Attack

ICMP (echo-requests and echo-replies) messages should ideally be used to verify a device on the network for connectivity between the sender and the destination device. This implies that an echo request packet may be sent to a network's broadcast address, and all hosts in the network will respond. The attacker uses a legitimate address of the victim while faking the source address, and all hosts on the network that receive the broadcasted echo request reply to it. This is illustrated in Fig 1.5

1.2 Background

Attackers continue to develop new processes and tactics for fooling defensive systems, allowing them to illegally use existing software and harm service providers. Emerging technologies, such as the Internet of Things (IoT), have recently been utilized to conduct strong and extremely successful DDoS assaults. Cisco Annual Internet Report (2018–2023) White Paper has predicted number of DDoS attacks will double to 15.4 million by 2023 globally. It said in its report that the frequency of breaches and the quantity of records exposed each breach are both increasing. Between 2018 and 2019, assaults between 100 Gbps and 400 Gbps increased by 776 percent Y/Y, and the overall number of DDoS attacks will double from 7.9 million in 2018 to 15.4 million by 2023.

For example, In February 2020, Amazon Web Services (AWS) was struck by a massive DDoS assault that lasted three days and peaked at 2.3 gigabytes per second. Also, multiple U.S. banks were attacked by DDoS assaults on March 12, 2012, which were carried out by Brobot and generated approximately 60 gigabits of DDoS attack traffic per second. Similarly, in 2017, Google's infrastructure was subjected to a 2.5Tbps

DDoS assault, the biggest of its kind in terms of volume [2].
A distributed denial of service (DDoS) assault can have disastrous consequences. For every hour that a successful DDoS assault lasts, the cost to a business is quite significant. There are couple of approaches that organizations adopt to prevent DDoS attacks, a lot of which are not effective.
More Bandwidth: Ensure that you have adequate bandwidth to manage surges in traffic generated by malicious activity as one approach to mitigate DDoS assaults by making your hosting infrastructure "DDoS resistant." In the past, you could protect yourself from DDoS assaults by ensuring that you had more bandwidth than any attacker was likely to have. However, with the emergence of amplification attacks, this strategy is no longer viable.
Network and Infrastructure Redundance: Making sure servers are distributed across many data centers with a robust load balancing system to transfer traffic between them to make it as difficult as possible for an attacker to conduct a DDoS attack against them. These data centers should be located in separate nations. For this method to be really effective, the data centers must be connected to several networks and there must be no visible network bottlenecks or single points of failure on these networks. It will be difficult for an attacker to successfully target more than a part of the servers if they are distributed geographically, leaving other servers untouched and capable of handling at least some of the increased traffic that the afflicted servers would typically manage. This will not entirely stop the attack, but it will limit the portion of the affected network or servers. This solution is also expensive.
Hardware Firewall: This may assist to avoid a DDoS attack by making a few basic hardware setup adjustments. Configuring the firewall or router to discard incoming ICMP packets or to block DNS replies from outside the network (by stopping UDP port 53), for example, can assist avoid some DNS and ping-based volumetric assaults. This method, however, can only stop some ICMP and UDP type of volumetric attacks.
Use of Machine Learning: Several machine learning approaches have been employed to identify DDoS attacks in recent years. The core idea underlying machine learning is to automatically learn from a set of data in order to spot patterns. Defense systems can employ machine learning approaches to identify whether a particular user is a regular user or an attacker. In the first stage, incoming network packets are analyzed and added to the database using filtering policies. Selected database features are extracted throughout the feature extraction procedure (feature selection process is discussed soon). Selected features are normalized to increase the training process' effectiveness.

The training is done via machine learning algorithms. The training step is carried out using machine learning algorithms, which learn patterns from the dataset. An arriving packet is classified as a DDoS attack, or a genuine user based on the learning parameters. The system then deletes the discovered DDoS packets and adjusts its filtering strategy to apply to fresh incoming traffic in the last step.
The method of choosing the best subset of features from an existing collection that contributes the most to the prediction performance is known as feature selection. This aims to increase the model's performance by removing unnecessary and irrelevant features that bear noise and reduce the model's accuracy. When constructing a machine learning algorithm, it is not necessary to utilize every feature available in the data. The algorithm for the DDoS detection can be improved by providing only the most critical features into it.
Various techniques of coping with DDoS assaults have been presented in previous research. To counter DDoS assaults, the bulk of these methods employ various machine learning techniques. Hybrid methods, which integrate two or more distinct machine learning algorithms, have also been presented. This research seeks to aid the research community in the creation and development of successful new DDoS attack solutions, since such solutions would help service providers minimize their risk of being hacked.

1.3 Problem Statement

To explore the efficacy of machine learning and deep learning-based algorithms for DDOS detection. When utilizing a conventional networking architecture, detecting DDoS attacks can be challenging since network information and control are decentralized. ML and DL method of detecting DDoS attacks seems to be the most reliable method particularly when it comes to correctly identifying non-malicious traffic. This differs from typical methods, which seek to classify harmful traffic while also wrongly classifying non-malicious packet, thereby degrading normal network traffic performance. This leads to the need to explore the effectiveness and efficiency of using machine learning and deep learning to detect DDoS attacks.
For this method of DDoS detection to be fully explored, different ML algorithms should be adopted in order to analyze and compare their results. Further to that, different feature selection techniques should also be implemented to get the best overall result. These will determine if ML-based approach is indeed the best methodology to detecting DDoS attacks.

1.4 State of the art

Khuphiran et al [3] compared Deep Feed Forward (DFF), a new developing deep learning method, to the conventional SVM. The efficacy of these two methods is evaluated using the DARPA Scalable Network Monitoring and DARPA 2009 DDoS assaults datasets. The dataset is preprocessed to see whether the classification process may be sped up.
From their experiment, the DFF deep learning system obtained a high accuracy of 99.63 percent in the trials, with a training time of 289.614 seconds. With a training time of 371.118 seconds, SVM obtained the maximum accuracy of 93.01 percent. They observed that SVM, on the other hand, can provide a quicker classification time. As a result, DFF is appropriate for situations when accuracy is paramount, but SVM is appropriate where classification speed is important.
Bakker et al. [4] investigated the efficacy of utilizing ML to identify DDoS assaults, which was made possible by Software-Defined Networking (SDN), a new paradigm aimed at improving network administration by centralizing network information and control. In their research, machine learning algorithms are built on nmeta2, an SDN-based traffic classification architecture, and tested on a physical network testbed to show their usefulness in a DDoS assault scenario, particularly in properly identifying non-malicious data.
This contrasts with most methods, which seek to identify harmful traffic while also misclassifying non-malicious data, resulting in good network traffic performance degradation. Furthermore, DDoS attacks can result in significant data loss, which can further decrease classification performance. They looked at the problems that occur when applying machine learning to identify DDoS assaults in real-world networks.
To identify and categorize distinct types of network traffic flows, Parvinder et al. [5] utilized a machine learning-based technique. The suggested method is tested on a fresh dataset that includes a combination of contemporary forms of attacks such as HTTP floods, SID DoS attempts, and regular traffic. WEKA, a machine learning technology, is used to classify different sorts of attacks. They utilized a dataset with twenty-seven features divided into five classes. The attacks in the dataset were classified using four different classifiers: J48, MLP, Random Forest, and Naive Bayes. The findings of four classifiers were examined, and it was discovered that the J48 classifier outperformed the other two. J48 had an accuracy of 98.64 percent, whereas the other three algorithms, MLP, Random Forest, and Naive Bayes, had accuracy of 98.63 percent, 98.10 percent, and 96.93 percent, respectively.

1.5 Contribution of Thesis

Several previous studies have looked at the use of ML and DL approaches in various areas, but the application of the learning to DDoS attacks is only getting started. As a result, this project work broadens the scope of prior research by concentrating on alternative machine learning classifiers for identifying DDoS attacks in modern networking settings. As a result, this work makes a number of significant contributions to this field of research:

- It offers up to seven different feature selection techniques of machine learning to determine the best subset of features from the data set in determining if the traffic is DDoS infected or not.
- It presents and analyzes ten separate machine learning classifiers to predict DDoS-infected traffic and their corresponding time to determine the most effective and fastest.It then compares the performance of supervised learning with semi-supervised learning using the same dataset.
- It categorizes and analyzes the finding of numerous researches on DDoS attack detection using various forms of ML and DL, as well as diverse assessment criteria. It also offers a brief overview of the many uses of machine learning and deep learning in fighting various forms of DDoS attacks.

1.6 Organization of Thesis

The following is the overall structure of this research work: The first chapter covers introduction where I illustrated what DDoS is and its classifications with relevant diagrams. I also gave background by talking about different approaches to prevent DDoS attacks and highlighted my contributions in the research area of DDoS attack detection. Chapter 2 analyzes and summarizes previous work done by other researchers with respect to using machine and deep learning to prevent DDoS attacks. Machine Learning and Deep Learning-based strategies for DDoS detection and introduction to the selection techniques used were discussed in chapter 3.
The role of feature selection in making ML predictions and the seven different feature selection techniques used in this paper were discussed in chapter 4 with the different visualization approaches used. Chapter 5 covers in detail all the ten different classifiers used to make DDoS predictions. Deep learning as an approach to predicting DDoS attack are also covered in this chapter. Chapter 6 summarizes the experimental results of using different feature selections techniques with the different ML classifiers, DL and semi-supervised learning.

Chapter 2

Literature Review

This chapter discusses the previous work done on DDoS detection. It talks about the literature reviews on the detection of the attack using Machine Learning (ML) and Deep Learning (DL). The first subsection analyses several researches done on DDoS assault detection using ML and the second subsection analyses researches on the detection using DL.

2.1 DDoS Detection Using ML

Priya et al. [6] used machine learning to develop an automated DDoS detector that can operate on any commodity hardware. The accuracy of the results is 98.5 percent. they utilized two parameters, delta time and packet size, to distinguish DDoS packets from regular packets using three classification algorithms: K-Nearest Neighbor, Random Forest, and Naïve Bayes. The detector can typically identify all forms of DDoS attacks, including ICMP floods, TCP floods, and UDP floods, among others.
Some systems may require a big number of characteristics to identify DDoS in older systems, while some systems may require many features to detect DDoS in older systems. Some systems may only function with specific protocols. Their suggested approach, on the other hand, overcomes these challenges by identifying DDoS of any form without the requirement for a specialized protocol that employs less characteristics.
Doshi et al. [7] were inspired to create new ways for detecting consumer IoT attack traffic automatically. They showed that leveraging IoT-specific network behaviors to guide feature selection can result in high accuracy DDoS detection in IoT network traffic using a range of machine learning methods, including neural networks. These findings suggest that utilizing low-cost machine learning techniques and flow-based, protocol-agnostic traffic data, home gateway routers or other network middleboxes

might automatically detect local IoT device sources of DDoS assaults.

Aysa et al. [8] applied pandemic modeling techniques to IoT networks made up of WSNs. They offered a framework for identifying and detecting aberrant defensive actions. There are substantial problems based on the influence of IoT-specific factors such inadequate processing capacity, power restrictions, and node density on the creation of a botnet.

They used standard datasets for two well-known assaults, including Mirai. To detect anomalous behaviors such as DDOS characteristics, they employed a variety of machine learning and data mining techniques such as LSVM, Neural Network, and Decision Tree. In the experiments, they discovered that combining a random forest with a decision tree resulted in a high level of accuracy in detecting attacks.

Ajeetha et al. [9] developed a unique approach for detecting DDoS attacks by analyzing traffic flow traces. These traces were used to create a confusion matrix. Using the normal and attack profiles acquired from existing datasets, two classifiers, Naïve Bayes and Random Forest, are utilized to categorize the traffic as abnormal or normal. The Naïve Bayes method outperforms the Random Forest approach.

Barati et al. [10] presented a detection system design for DDoS attacks. In their hybrid technique, the Genetic Algorithm (GA) and Artificial Neural Network (ANN) are used for feature selection and attack detection, respectively. The most efficient features are selected using a wrapper approach based on GA, and the detection rate of DDoS attacks is enhanced using ANN's Multi-Layer Perceptron (MLP). The results show that the suggested approach can identify DDoS attacks with high accuracy while avoiding False Alarms.

Sudugala et al. [11] offered a DDoS detection technique based on Machine Learning that has increased accuracy and reduced false positive rates. The suggested method derives inferences from signatures previously collected from network traffic samples. They used four different benchmark datasets and four different machine learning techniques to handle four of the most dangerous DDoS attack routes. The authors obtained the highest level of accuracy and compared their findings to those of other machine learning methods.

Zecheng et al. [12] suggested a cloud-based DOS attack detection system based on machine learning techniques on the source side. To prevent network packages from being pushed out to the outside network, this solution uses statistical information from both the cloud server's hypervisor and the virtual machines.

They compared the performance of nine different machine learning algorithms. According to their findings, more than 99.7 percent of four different types of DOS assaults are effectively identified. Their method has no negative effects on performance

and may be readily adapted to other types of DOS attacks.
Luong et al. [13] presented a machine learning (ML) and deep neural network (DNN)-based DDoS detection and prevention technique in Software Defined Network (SDN) systems. The use of ML and DNN classifiers in conjunction with SDN's centralized features may effectively reduce the network system's DDoS damage. Aside from that, they ran two sorts of attack scenarios, one from within the network system and the other from outside.
The online monitoring system (OMS), spoofed traffic detection module, and interface-based rate limitation (IBRL) algorithm are all part of Devi et al. [14] proposed concept. By monitoring the degradation in host and network performance indicators, OMS delivers real-time DDoS impact measurements.
The spoofed traffic detection module uses the hop count inspection algorithm (HCF) to verify the authenticity of incoming packets using the source IP address and subsequent hops to the intended target. HCF in combination with a support vector machine (SVM) has a 98.99 percent accuracy rate with fewer false positives. Following that, when system limitations are exceeded, the IBRL algorithm reduces traffic aggregation at the victim router in order to give sufficient bandwidth for remaining flows
Wehbi et al. [15] surveyed. They attempted to highlight some of the most recent Machine Learning (ML) techniques for detecting DDoS assaults in IoT networks, as well as their benefits and drawbacks. A comparison of the results of several techniques is also presented.
Zekri et al. [16] developed a system to combat the DDoS threat. They developed a DDoS detection system based on the C.4.5 algorithm. This method, when combined with signature detection approaches, creates a decision tree that can identify signature attacks for DDoS assaults automatically and effectively. They chose different machine learning approaches and compared the results to validate our system.

2.2 DDoS Detection Using DL

Hussain et al. [17] developed a unified framework for early detection of distributed denial-of-service (DDoS) attacks organized by a botnet that controls malicious devices, based on deep convolutional neural networks (CNNs) and actual network data. Individually, these puppet devices conduct quiet call, signaling, SMS spamming, or a combination of these assaults against call, Internet, SMS, or a combination of these services to produce a collective DDoS attack in a cell that disrupts CPS operations. Their findings show that their framework can recognize normal and under-attack cells

with an accuracy of more than 91 percent.

Doriguzzi-Corin et al. [18] demonstrated Lucid, a viable, lightweight deep learning DDoS detection solution that uses the characteristics of Convolutional Neural Networks (CNNs) to categorize traffic flows as malicious or benign. The following contributions were made by them: To begin, a novel use of a CNN to identify DDoS activity with little processing overhead. Second, a dataset-independent preprocessing method to generate traffic observations for online attack detection, followed by an activation analysis to explain Lucid's DDoS categorization, and last, empirical validation of the methodology on a resource-constrained hardware environment. They demonstrated that the suggested technique is suitable for successful DDoS detection in resource-constrained operational situations using their assessment findings. Wang et al. [19] proposed a DDoS attack detection technique based on deep learning and information entropy. To begin, the controller can analyze suspicious traffic using information entropy detection. The convolutional neural network (CNN) model then performs fine-grained packet-based detection to discriminate between regular and attack data.

Finally, the controller executes a defensive strategy in order to thwart the attack. Their experiments show that the technique has a 98.98 percent accuracy rate, indicating that it has the ability to successfully detect DDoS attack traffic in an SDN context.

Ma et al. [20] demonstrated a unique deep learning approach for detecting DDoS. They described the suggested convolution neural network (CNN) model, which is based on the feature fusion process that they created. A Symmetric logarithmic loss function based on categorical cross entropy was also presented.

The suggested detection system has also been implemented to GPU-enabled TensorFlow and tested using the NSL-KDD datasets as a benchmark. Extensive testing shows that the proposed model outperforms existing techniques and has a lot of promise for use in IoT threat detection.

DDoSNet, an intrusion detection system for DDoS assaults in SDN settings, was proposed by Elsayed et al. [21]. The Recurrent Neural Network (RNN) and autoencoder are used in our method, which is based on Deep Learning (DL). They tested the model with the released dataset CICDDoS2019, which covers a wide range of DDoS attacks and fills in the gaps in previous datasets. When compared to previous benchmarking approaches, they saw a substantial improvement in attack detection. As a result, their methodology gives them a lot of confidence when it comes to protecting these networks.

Wang et al. [22] presented a software-defined Internet of Things (IoT) framework

(SD-IoT). A SD-IoT controller, SD-IoT switches coupled with an IoT gateway, and IoT devices make up the proposed architecture. Using the suggested SD-IoT architecture, they then present a deep learning detection technique based on time series. Finally, findings from experiments demonstrate that the suggested method performs well.

He et al. [23] proposed a deep transfer learning-based small-sample DDoS assault detection technique. To begin, deep learning techniques are utilized to train multiple neural networks that may be used for DDoS attack transmission given enough data. Then, to compare the transfer performance of different networks, they create a transferability metric.

The network with the best transfer performance may be chosen among the four networks using this criterion. They then proved that the deep learning detection approach degrades performance for a small sample of DDoS assaults, with detection performance decreasing from 99.28 percent to 67 percent. Finally, deep transfer of the 8LANN network in the target domain resulted in a 20.8 percent improvement in detection performance. The experiment indicates that the suggested detection approach based on deep transfer learning may significantly improve the performance degradation of deep learning techniques for detecting DDoS attacks with few samples.

Bhati et al. [24] proposed a Deep Neural Network architecture. High-level features are retrieved for representation and inference of the dataset in their method. The experiment was carried out using the ISCX datasets for the years 2017, 2018, and CICDDoS2019, and the software was written in Matlab R17b using Wireshark.

Chapter 3

Machine Learning and Deep Learning Based Strategies for DDoS Detection

DDoS (Distributed Denial of Service) assaults have emerged as a serious danger to the security and integrity of computer networks and information systems, which are critical modern infrastructures. Detecting DDoS assaults is a difficult task that must be completed before any control measures can be implemented. The application of machine learning and deep learning to the detection of DDoS assaults has yielded positive results.

3.1 Model Architecture

The model takes data as its input and have predicted data as its output. The data is divided into two sections. The first portion is referred to as training data, while the second is referred to as testing data. From the training data supplied to it, the model learns to execute a certain task - prediction. After the model has been trained, the projected values are compared against real values from the testing data to see if the model's estimates are right. These datasets are not in the ideal forms for data mining techniques to be used on them so before processing, data transformation, discretization and normalization method was used to transform the raw data into a more usable format for next set of algorithms, which are feature selections and classifiers as shown in Fig 3.1.

During training of the system, the transformed data is first fed into different feature selection algorithms to determine the best and relevant feature sets before passing

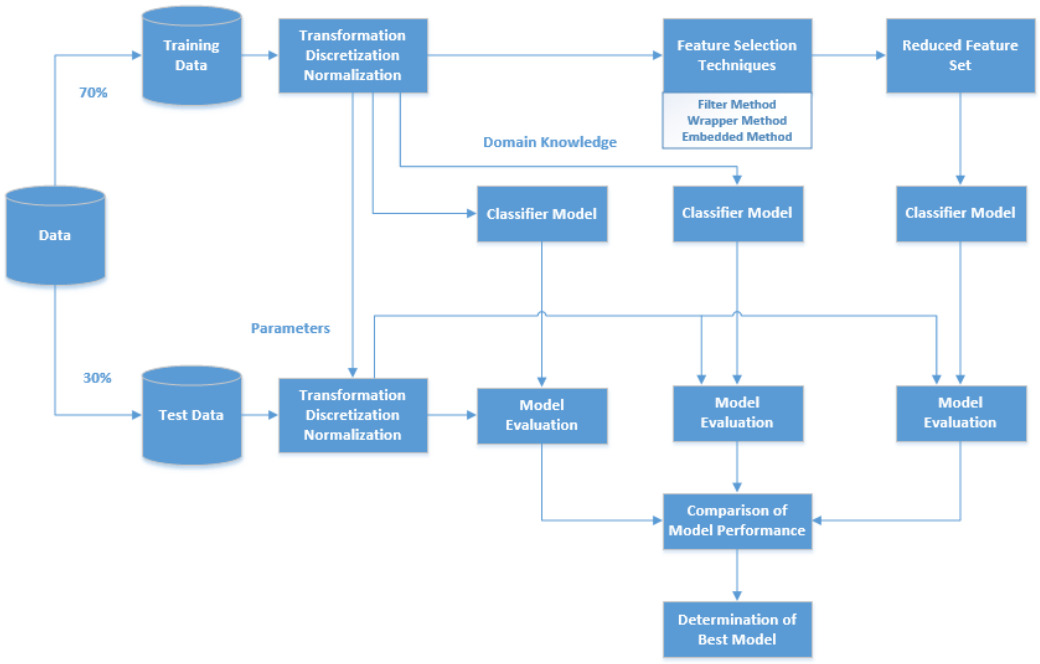

Figure 3.1: Model Architecture for DDoS attack detection

into the classifier. Secondly, the transformed data is passed into the classifier model using domain knowledge and lastly directly into the classifier model. The output from the three classifiers were assessed using three different evaluation models. The models were then compared to determine the best.

3.2 Dataset Generation

Data can be generated in several ways and producing a credible background traffic is a top priority in building dataset. Some popular tools used in capturing traffic on the network for DDoS attack types are Metasploit, Nmap, and Wireshark. Wireshark, been the most popular and free network protocol analyzer. It gives a tiny view of what is going on the network. It has several capture file formats like Cisco Secure IDS iplog, Microsoft Network Monitor, tcpdump, Pcap etc. Wireshark also has the ability to export the output file to CSV, XML, or plain text [25]. It is also possible to feed in Wireshark output to a model (written in python). These good features make it a great tool for capturing data to be used for DDoS attack detection.
Datasets must be split into training and testing. The model is trained using the training set and tested using the testing set. The main goal is to get an unbiased assessment of how well the machine learning model functions using a test dataset. The

performance metric will have a significant variance if the test dataset is too small and train dataset too large. There will also not be enough training data to construct a well-performing model if the test dataset is too large. For a moderately sized dataset, the most adopted splits are 70/30 and 80/20, the higher being training and smaller testing. When machine learning algorithms are employed to estimate predictions on data, the train-test split process is used to assess their performance.

3.3 Data Pre-processing

There is need for preprocessing of the data which entails converting raw data into a format that can be understood. Most data are frequently inadequate, inconsistent, and/or missing in specific behaviors or patterns. Real-world data is generally of poor quality and cannot be easily fed into ML models. In addition, such data is mostly incomplete. As a result, it may be difficult to uncover hidden features that are of interest to the domain expert, or the data may contain mistakes. The quality of the input data has a significant impact on the inference and analysis results produced by ML models. In the absence of high-quality data, a highly successful approach is typically incapable of generating accurate conclusions. The dataset used in this research work was transformed, discretized, and normalized before feeding into different categories of classifiers.

3.3.1 Data Transformation

This is the process of transforming data from one format or structure to another. It's used to make data more consistent with the assumptions of a statistical inference method, or to make data more interpretable. It transforms data from one domain to another using the $y = f(x)$ transformation function [26]. Switching numeric features into non-numeric or vice versa, converting the string to some kinds of numeric representation, inputs are resized to a set size are all examples of data transformation. Transformation was carried out prior to training in this study. The machine learning model and the code are segmented.

3.3.2 Data Discretization

Depending on the experimental set-up, existing conditions, and variables of interest, data collected from various sources is available in various forms. It is often more convenient and beneficial to discretize data in order to evaluate it. Discretization divides each quantitative attribute's value domain into distinct intervals. Using discrete values while processing data has a lot of advantages. These benefits include better performance, better understanding of data and easier to visualize, and consumes less memory [27]. Supervised data discretization was adopted in this paper where class labels are used to transform continuous data into discrete ranges generated from the original dataset.

3.3.3 Data Normalization

This is the process of transforming data such that it is generally normally distributed. If the model has any specified assumptions, the data should be normalized. It is a scaling method that shifts and rescales data to make them range between 0 and 1. Min-Max normalization a popular type of normalization where it converts the value of an attribute in the original dataset, say x, to a new value, say x' [26]. Other methods of data normalization are Z-score, decimal scaling and quantile normalization. Min-Max normalization was used for normalizing the dataset in this research work.

3.3.4 Domain Knowledge

Simply said, this is information regarding the model's working environment. It may be possible to decrease the burden of data gathering and/or acquire data of higher quality if suitable domain insights are available. One option for removing duplication is to include domain-dependent information from knowledge bases into the data cleansing operation. With the application of such data knowledge, the model's accuracy improves.

3.4 Feature Selection Techniques

It is mostly believed that Machine learning prediction is based on how good your algorithm is or how robust is the computer running it. In most situations, it is found that there are two things that differentiate the good ones from the rest - the data

and the selection of features (input variable). When there are a lot of features, this becomes much more significant. When constructing this algorithm, all features available were not utilized. It has helped the algorithm by providing only the most critical features into it helping the DDoS detection model to be more accurate. There are 3 basic feature selection strategies employed in this research work.

3.4.1 Filter-based Methods

This technique is more versatile and less computationally expensive than wrapper and embedded techniques since they are independent of the supervised learning method. Filter techniques, rather than wrapper and embedding techniques, are appropriate for processing high-dimensional data. The goal of the feature selection procedure is to identify the most relevant features.

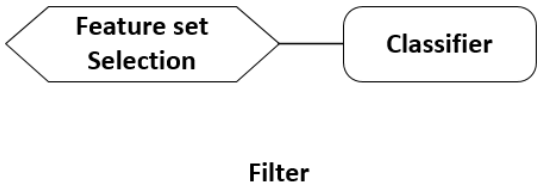

Figure 3.2: Filter Model

3.4.2 Wrapper-based methods

The wrapper-based method, one of the techniques adopted in this paper, builds feature subsets using any of the searching strategies and then assesses them using a supervised learning algorithm in terms of classification error or accuracy [27]. The wrapping approach appears to be "brute force." For picking the relevant features from the dataset, Kohavi and John devised a wrapper-based feature selection approach [28]. This approach uses a search engine to generate subsets and a classification algorithm to assess them. Fig 3.3 below shows how the classifier is wrapped into the feature selection method.

3.4.3 Embedded-based methods

This approach utilizes a portion of the supervised learning algorithm's learning process. Embedded-based techniques are less expensive to compute than wrapper techniques [29]. Pruning technique, built-in mechanism, and regularization models are

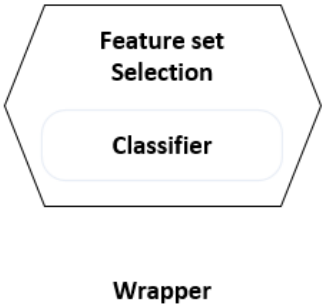

Figure 3.3: Wrapper Model

the three types of embedded methods that may be found. Fig 3.4 shows how the feature selection technique is embedded into the classifier for this DDoS detection.

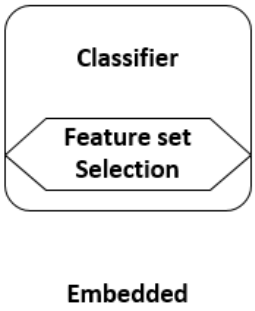

Figure 3.4: Embedded Model

3.5 Classifier Models

Machine and deep learning research have exploded in the past couple of years. These techniques are being used to solve new types of problems, such as database knowledge discovery, language processing, robotic control, and optimization models, as well as more traditional problems like speech recognition, facial recognition, handwriting recognition, game playing, medical data analysis as well as attack detect like DDoS. An ML or DL model is simply a computer program that has been taught to identify specific patterns. The model is trained on a collection of data called the dataset and giving it an algorithm to use to make predictions from that data. Once the model has been trained, it can be used to reason over data it has not seen before and make meaningful predictions about it. That is why the DDoS prediction model in this research was used to predict if a DDoS attack is imminent or not by identifying if there is TCP, UPD, ICMP flood attack. There are ten different MLs, one DL and one semi-supervised learning adopted for DDoS prediction in this paper.

Chapter 4

Feature Selection and Visualization

Understanding the most significant characteristics to employ is essential for creating a successful model. Experimentation is required to determine which qualities to examine, and appropriate presentation of the data can assist in clarifying the first choices.

4.1 Feature Selection

This is the process of selecting variables that are beneficial in predicting a response in machine learning. When creating predictive models, it is a good idea to figure out which attributes are significant. There are three major methods for selecting features. Filter methods, Embedded methods, and Wrapper methods. The differences between the 3 major feature selection techniques are summarized in the Table 4.1 [30].

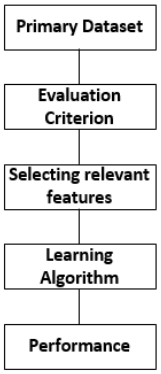

Figure 4.1: Filter-based method of feature selection

Table 4.1: Major differences between 3 feature selection techniques

Filter	**Wrapper**	**Embedded**
A collection of approaches that do not use a particular machine learning algorithm.	To discover the best characteristics, it evaluates a certain machine learning algorithm.	During the model construction phase, embeds features. During the model training phase, each iteration of the feature selection is observed.
In terms of time complexity, it is far quicker than wrapper techniques.	For a dataset with many features, the calculation time is long.	In terms of time complexity, it falls between filter methods and wrapper methods.
Overfitting is less likely.	Because it requires training machine learning models with diverse combinations of features, there is a high risk of overfitting.	Used to generally minimize overfitting by penalizing models with very high coefficients.
Examples include ANOVA, Variance Threshold, Mutual Information, Correlation etc.	Examples include Forward selection, backward selection, Bi-directional etc.	Examples include LASSO, Ridge etc.

4.1.1 Filter method

Filter methods compute the relationship between features and the target variable using mathematical techniques. To rate the significance of individual functions, filter approaches often use statistical test scores and variances. Filter methods are not dependent of the machine learning algorithms. Fig 4.1 illustrates this method. As a result, they may be utilized as input to any machine learning model and are extremely quick. Instead of cross-validation performance, it measures the intrinsic qualities of the features using univariate statistics. These approaches are more efficient and cost less to compute.

Five examples of filtering methods adopted for this DDoS attack detection model:

- **ANOVA F-value:** This calculates the degree of linearity between the input features (independent features) and output (dependent feature). A high F-value implies that the degree of linearity is strong, whereas a low F-value suggests that the degree of linearity is low. However, ANOVA F-value only captures the linear relationships between the two categories of feature.
- **Variance Threshold:** This assign threshold values to the features and gets rid of the features whose variance is less than the set threshold. It removes all zero-variance features by default, i.e., features with the same value across all samples. It can be used for both supervised and unsupervised learning. It simply looks at the relationships between the features, not the relationships between the

input and output features. The features with a greater variance contain more important information, but the drawback is that it does not take into account the link between the feature variables and the target variables.

- **Mutual information:** It measures the quantity of information gained about one feature through the other feature to measure the dependency of one variable on another. It is symmetric and non-negative, and it equals zero if and only if two random variables are independent; higher values indicate greater dependence. It can handle non-linear relationships between input and output features. Mutual information in machine learning refers to the amount of information that the presence or absence of a feature provides to making the right prediction on Y. Their MI is zero if X and Y are independent. MI is the entropy of X, which is a concept in information theory that assesses or quantifies the amount of information within a variable if X is deterministic of Y. It can be mathematically represented as shown in the equation below [31].

$$I(X;Y) = \sum_{x,y} P_{XY}(x,y)\, log\left[\frac{P_{XY}(x,y)}{P_X(x)\, P_Y(y)}\right]$$

- **SelectKBest:** This "removes everything but the k highest scoring features" after selecting the features with a function (this case, ANOVA F-value).

- **Pearson's Correlation:** A statistic that calculates the linear correlation between two continuous variables. It ranges from -1 to +1, with +1 indicating positive linear correlation, 0 indicating no linear correlation, and 1 indicating negative linear correlation. It is a well-known metric in the field of machine learning. It is a metric for expressing the strength of a linear relationship between two variables. It can be mathematically represented as shown in the equation below

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

  Where x_i are features, y_i are labels, $\bar{x}$ and $\bar{y}$ are the mean.

4.1.2 Wrapper Method

This strategy is based on greedy search algorithms, which analyze all potential feature combinations and choose the one that delivers the best result for a given machine learning algorithm. One disadvantage of this approach is that it might be computationally expensive to test all potential combinations of the characteristics, especially if the feature collection is big. Because they must use the learning algorithm many

times to discover the best performing subset of characteristics, this is obviously a computationally costly strategy. This method is illustrated in Fig 4.2.

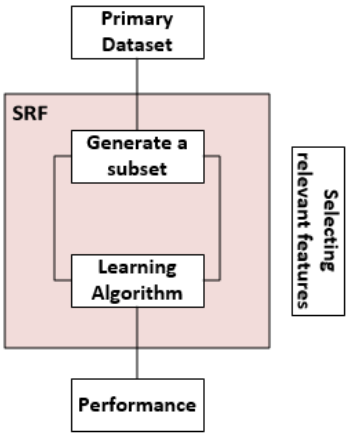

Figure 4.2: Wrapper-based method of feature selection

The following are the steps for using wrapper approach [30]. High level steps of wrapper-based method of feature selection. Input: Dataset D, Target T Output: Selected features S

1. Initial set of all features from Dataset D
2. Generate/consider a subset of features
3. ML Algorithm
4. Gauge model performance
5. Repeat steps 2-4 until optimal set of features are gotten
6. Selected features

There are four main examples of wrapper method:

- **Forward Selection:**This is an iterative strategy in which the model starts with no features. We keep adding the feature that best improves the model in each iteration until adding a new variable no longer enhances the model's performance.
- **Backward Elimination:** Backward elimination begins with all the features and eliminates the least significant feature at each iteration, thus improving the model's performance. This process is repeated until no improvement is noticed when features are removed.
- **Bi-Directional Elimination:** It is similar to forward selection, but instead of adding a new feature, it evaluates the relevance of previously added features, and

if any of the previously picked features are found to be unimportant, it simply removes that feature by backward elimination. The following are the steps for using the Bi-Directional selection approach [32].

Algorithm

Input: Dataset D, Target T
Output: Selected Variables S
1. $S \rightarrow \phi$ //Set of selected variables
2. $R \rightarrow V$ //Set of remaining candidate variables
3. //Forward phase: iterate until S does not change
4. while S changes do
5. //Identify the best variable V_{best} out of all remaining variables R, according to $Perf$
6. $V_{best} \rightarrow argmax_{V \in R} Perf(S \cup V)$
7. //Select V_{best} if it increases performance according to criterion C
8. if $Perf(S \cup V_{best}) >_c Perf(S)$ then
9. $S \rightarrow S \cup V_{best}$
10. $R \rightarrow R/V_{best}$
11. end if
12. end while
13. //Backward phase: iterate until S does not change
14. while S changes do
15. //Identify the worst variable V_{worst} out of all selected variables S, according to $Perf$
16. $V_{worst} \rightarrow argmax_{V \in S} Perf(S/V)$
17. //Remove V_{worst} if it does not decrease performance according to criterion C
28. if $Perf(S/V_{worst}) \geq_c Perf(S)$ then
19. $S \rightarrow S/V_{worst}$
20. end if
21. end while
22. return S

$Perf$ examines a collection of variables and returns their performance in comparison to a statistical model. C is the selection criterion, V is variable, V_{best} is best variable, and R is the remaining variable.

- **Recursive Feature Elimination (RFE):** This is a greedy optimization method that seeks for the best-performing feature subset. This method was adopted in this research work. It produces models over and over again, putting away the best or worst performing feature at each iteration. It builds the next model using the features set aside until all the features are used up. The features are then ranked in order of their deletion.

4.1.3 Embedded Method

This approach combines the benefits of both the filter and wrapper methods. It is implemented by algorithms that have their own built-in feature selection methods. These approaches complete the feature selection process inside the machine learning algorithm's design. In other words, they choose features during model training, which is why they are referred to as embedded methods. The learning algorithm uses its own variable selection mechanism to simultaneously conduct feature selection and classification/regression. This method is illustrated in Fig 4.3.

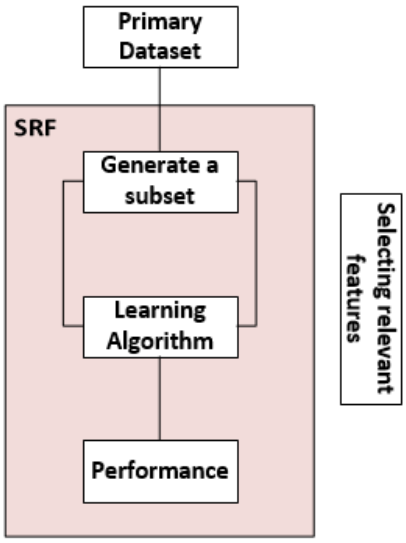

Figure 4.3: Embedded-based method of feature selection

There are two different types of embedded feature selection method:

Regularization Method

This reduces the freedom of a model by imposing a penalty on its many parameters. This penalty is applied to the coefficient that multiplies each of the features in the linear model, and it is done to reduce overfitting, improve generalization, and make

the model resilient to noise. There are two main types of regularization for linear models:

- **LASSO Regression:** It is also called an L1 regularization. It reduces some coefficients to zero, implying that a certain predictor or set of features will be multiplied by zero to estimate the target. As a result, it will not be included in the target's final forecast — this indicates that these features can be eliminated because they do not contribute to the final prediction. Lasso is best used when there is a large number of features, which works well for feature selection. The RSS plus the sum of absolute values of weight magnitudes is the objective function (also known as the cost) to be minimized. Numerically, this may be expressed as shown below [33].

  $Cost(W) = RSS(W) + \lambda * (sum of absolute values of weights)$

  $$= \sum_{i=1}^{N} \left[y_i - \sum_{j=0}^{M} w_j x_{ij} \right]^2 + \lambda \sum_{j=0}^{M} |w_j|$$

  y is the label, x is the feature and w is the weight.
  The gradient is not determined in this situation because the absolute function is not distinguishable at $x = 0$.

  The following methods are used in one of the coordinate descent algorithms.
  Step 1 - initialize weights w(by equating w to 0)
  Step 2 - iterate till not converged
  iterate over all features from $j = 0$ to M
  update the j^{th} weight with a value which minimizes the cost
  Step 3 end

- **RIDGE Regression:** It is also called L2 regularization. This, on the other hand, does not set the coefficient to zero, but rather just gets close to it, which is why we only employ L1 in feature selection. The RSS plus the sum of squares of the magnitude of weights is the objective function (also known as the cost) to be minimized. This may be expressed numerically as follows [33]:

  $Cost(W) = RSS(W) + \lambda * (sum of squares of weights)$

$$= \sum_{i=1}^{N} \left[y_i - \sum_{j=0}^{M} w_j x_{ij} \right]^2 + \lambda \sum_{j=0}^{M} w_j^2$$

Where y is the label, x is the feature and w is the weight. The gradient value will then be

$$\frac{\partial}{\partial w_j} Cost\,(w) = -2 \sum_{i=1}^{N} x_{ij} \left[y_i - \sum_{k=0}^{M} w_k x_{ik} + 2\ \lambda w_j \right]$$

Only w_j remains in the regularization section of gradient, and all others would become zero.
It is identical to that of basic linear regression. As a result, ridge regression is the same as lowering the weight by a factor of $(1 - 2\lambda\eta)$ and then using the same update procedure as basic linear regression.

Tree-based Methods

Tree-based algorithms and models (random forest) are well-known algorithms that can give us with what we term feature importance as a technique to choose features in addition to high prediction performance. Using simple approaches such as mean decrease impurity and mean decrease accuracy, random forests offer us with feature relevance. A random forest is a collection of decision trees, each of which is built using a random sampling and feature extraction from the dataset, so no one tree can see all of the features or access all of the observations. A decision tree's nodes are conditioned on a single feature. These nodes are intended to divide the dataset into two halves. Observation values that are similar will be in the same set, while those that are different will be in the other. As a result, the value of each feature is determined by how "pure" each collection is.

4.1.4 Bayesian Method

Bayesian method is a complete separate approach. It can be used as filter, wrapper or embedded method depending on how it is configured. When we have several competing models and need to choose the best one, Bayesian model selection (BMS) might be used.
BMS was created largely for the purpose of model comparison. However, it may

also be used as a feature selection tool [34]. If a linear regression models are created for all possible combinations of characteristics, the model with the lowest Bayesian Information Criteria (BIC) has all of the important features. This type of model selection framework was also employed in this paper to compare other feature selection techniques to determine the best algorithm for detecting DDoS attack.
It may significantly reduce the number of features in models while simultaneously making them more interpretable. Furthermore, it selects a basic linear regression model as a baseline model before introducing a non-linear machine learning model to train the data, which may be utilized to improve model performance.

4.2 Visualization

A lot of information is hidden behind data, as well issues in seeing the data's structure. There is need to comprehend data trends and patterns, assess data frequency and other features, determine the distribution of variables in the data, and finally, to display the link that may exist between various variables.

4.2.1 RadViz

This is a multivariate data visualization technique that shows each feature dimension evenly around the circumference of a circle, then depicts points on the inside of the circle, normalizing their values on the axes from the center to each arc. This technique permits as many dimensions as can fit on a circle, substantially increasing the visualization's dimensionality. Because they employ the familiar concept of 2D points for storing data components and displaying the original data dimensions that function as springs for setting the x and y coordinates, RadViz charts are widely used to depict multidimensional data [35].
By projecting an N-dimensional data set into a simple 2D environment where the effect of each dimension may be read as a balance between the influence of all dimensions, Radviz provides an easy way to visualization. A dimensional anchor represents each dimension in the dataset, and each dimensional anchor is uniformly distributed over a unit circle. A spring connects each dimensional anchor to each line in the data set, which corresponds to a point in the projection.
When using Radviz to visualize the DDoS data, classes of 'normal', 'udp', 'tcp', 'icmp' are created. These classes become the data points inside the circle with each classes having different colors.

4.2.2 Parallel Coordinates

Parallel coordinates are a popular method for viewing and analyzing large datasets. A background is drawn consisting of n parallel lines, usually vertical and evenly spaced, to depict a group of points in an n-dimensional space. Multivariate numerical data is plotted using this style of visualization. Parallel Coordinates Plots are useful for comparing several variables and visualizing their relationships.
Parallel coordinates are a typical technique of displaying and interpreting multivariate data in high-dimensional geometry. Each variable has its own axis in a Parallel Coordinates Plot, and all of the axes are parallel to each other. Because each variable uses a distinct unit of measurement, each axis can have a separate scale, or all the axes can be normalized to keep all the scales uniform. Many features of a multivariate data collection may be analyzed using a design with polylines depicting multivariate items intersecting with parallel axes representing variables [36].

Chapter 5

ML Algorithms

Machine learning are algorithms that improve themselves over time by gaining experience and using data. These algorithms are used to predict if a DDoS attack will happen or not. The type of algorithms used in this research are supervised, semi-supervised and Deep learning.

5.1 Supervised Learning algorithms

This method uses ML algorithms to learn from a labeled input variables in order to predict the output variables. When the algorithm reaches a satisfactory level of performance, the learning process comes to an end. Examples of supervised learning algorithms include k-nearest neighbor (KNN), linear regression, support vector machine (SVM), logistic regression, naive Bayes algorithms etc. Part of this research work made use of ten supervised machine learning algorithm to predict the class of the given data points. The supervised ML classifiers used are described briefly below.

5.1.1 K-Nearest Neighbor

It is a simple and easily applicable supervised machine learning algorithm that can be used for regression and classification. The k-Nearest-Neighbors (kNN) technique is a basic yet successful classification method. In many instances, the k-Nearest-Neighbors (kNN) technique is a non-parametric classification method that is simple but successful. To classify an x number of data records, the k closest neighbors are collected, forming a neighborhood of x. The categorization for x is generally decided by majority vote among the data records in the neighborhood, with or without consideration of distance-based weighting [37].
A distance metric is used to identify which of the K examples in the training dataset are most similar to a new input. The most prevalent distance metric for real-valued

input variables is Euclidean distance calculated as

$$\text{Euclidean Distance } (x, x_i) = \sqrt{\sum_{i=1}^{n} (x_j - x_{lj})^2}$$

k being a positive integer, KNN uses nearest k neighbors to determine the class of the new data point. However, to be successful, we must pick an acceptable value for k, and the classification success is highly reliant on this number. Although, throughout the model development process, the value of k can be made automated.
kNN is a case-based lazy learning technique for classification that maintains all the training data. It is a basic yet efficient categorization approach. It mostly uses Euclidean distance function to compute the nearest neighbor. When a new data comes in, Euclidean function is used to calculate the distance between this new data and data in the training set separately. K smallest distance (neighbors) is then selected to determine the class of the new data.

5.1.2 Gradient Boosting

This is a greedy algorithm and can overfit a training dataset quickly. It is a technique that can be used for both regression and classification issues. It produces a prediction model from several weak prediction models.
Gradient boosting is a learning method that fits new models in a sequence to produce a more accurate response variable estimate. The basic concept behind this approach is to build new base-learners that are maximally correlated with the loss function's negative gradient, which relates to the entire ensemble. The cost function can be calculated as

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y})^2$$

MSE is mean square error, N is number of points, Y_i is actual output and $\hat{Y}_i$ is the predicted output value.

The fundamental idea behind boosting is to incrementally add new models to the ensemble. A new weak, base-learner model is trained with regard to the error of the entire ensemble learned so far at each iteration. It gives the model architect a lot of leeway, making the selection of the best loss function a question of trial and error. Boosting algorithms, on the other hand, are very simple to implement, allowing for experimentation with various model designs [38].

Furthermore, the Gradient Boosting Machines have demonstrated significant success in a variety of machine learning and data mining issues, in addition to practical applications.

5.1.3 AdaBoost

This machine learning algorithm works by attaching weights to the observations, putting more weight on difficult to classify instances and less weight on those that were already handled well. New weak learners are added serially that focus their training on those more difficult patterns. There are two major issues with boosting: how to adapt the training set so that the weak classifier can train on it, and how to merge the weak classifiers obtained via training into a strong one.
Freund and Schapire created the Adaboost (adaptive boosting) method in 1995 to address the aforementioned issues, which works by changing weight without requiring any prior knowledge of learner learning. The algorithm could change voting-weights and solved many practical problems of the early Boosting algorithm. The AdaBoost method is mostly used to investigate and solve classification issues. It can handle problems like two-class problems, multi-class single label problems, multi-classification and multi-label problems, class single label problems, and regression problems, for example. It offers the advantages of being fast, simple to use, and simple to program. Except for the number of iterations, no other parameters need to be changed [39]. The loss function (I) can be represented as

$$I(f_m(x), y) = \{^{0 if f_m(x_i) \neq y_i}_{1 if f_m(x_i) = y_i}$$

where $x_i \in R^K$ and $y_i \in \{-1, 1\}$ denoted $f_m(x) \in \{-1, 1\}$

It may be coupled with any approach to find weak hypotheses, and it does not require any prior knowledge of WeakLearn. Given enough data and a WeakLearn with only somewhat dependable accuracy, it can give a set of theoretical learning guarantees.

5.1.4 Support Vector Machine

SVM is also a supervised machine learning algorithm which can be used for both classification and regression problems. It is however, commonly used for classification. It uses hyperplane in an N-dimensional space (N is the number of features) that clearly classifies the data points.

With roots in Statistical Learning Theory (SLT) and optimization methods, support vector machines have evolved into powerful tools for problem solving in machine learning with finite training points, overcoming some traditional challenges such as the "curse of dimensionality," "over-fitting," and so on. Theoretical foundations and implementation techniques for SVMs have been established, and SVMs are rapidly gaining popularity and development due to a number of appealing features, including enticing mathematical representations, geometrical explanations, good generalization abilities, and promising empirical performance [40].
SVM cost function can be written as

$$Cost(h_\theta(x), y) = \{^{max(0,1-\theta^T x) if y=1}_{max(0,1+\theta^T x) if y=0}$$

$$J(\theta) = \sum_{i=1}^{m} y_i Cost_1(\theta^T(x_i)) + (1 - y_i) Cost_0(\theta^T(x_i))$$

$$J(\theta) = \sum_{i=1}^{m} y_i max(0, 1 - \theta^T x) + (1 - y_i) max(0, 1 + \theta^T x)$$

m is number of samples

Many SVM algorithms solve non-convex and more general optimization problems, such as integer programming, semi-infinite programming, bi-level programming, and so on, as well as convex problems like linear programming, quadratic programming, second order cone programming, and semi-definite programming.

5.1.5 Naive Bayes

This is a simple probabilistic classifier based on Bayes' Theorem which is useful for large dataset. Naive Bayes model is easy to build when the features in the datasets are independent of each other. The classifier is fast and not sensitive to unrelated features. The Naïve Bayes performs very well in binary cases for example when the classification purpose is to discriminate if the incoming packets are DDoS or normal. The model learns by computing the probability of the training data.
By assuming that characteristics are independent of class, the naive Bayes classifier substantially simplifies learning. The degree of feature dependencies defined as the class conditional mutual information between the features has no direct relationship with the accuracy of naive Bayes. Instead, the quantity of information about the class that is lost due to the independence assumption is a stronger predictor of naive Bayes correctness [41]. The most common loss function for Naïve bayes is Residual sum of squares (RSS) calculated

$$RSS(\theta) = \sum_n (y_n - f(x_n|\theta))^2$$

Many practical uses of Naive Bayes have been demonstrated, including text categorization, medical diagnosis, and system performance management. In two situations, Naive Bayes performs best: fully independent features (as predicted) and functionally dependent features (which is surprising). Between these two extremes, Naive Bayes has the lowest performance. Surprisingly, the degree of feature dependencies assessed as the class-conditional mutual information between the features has no clear correlation with the accuracy of naive Bayes. Instead, the loss of information that features hold about the class when using a naïve Bayes model is a stronger predictor of accuracy [41].

5.1.6 Neural Network

This algorithm uses a basic building block called neurons. The collections of these connected neurons are called artificial neurons. ANN is a strong classification tool based on the artificial neuron model. Artificial neurons are designed to behave similarly to biological neurons in the biological brain.
The use of neural networks as a classification technique has become increasingly popular. Recent extensive research in neural classification has demonstrated that neural networks offer a potential alternative to a variety of traditional classification methods. The following theoretical characteristics of neural networks are advantageous.
First, neural networks are data-driven self-adaptive techniques in the sense that they may adjust to the data without any explicit definition of the underlying model's functional or distributional structure. Second, neural networks are universal functional approximators in the sense that they can estimate any function with arbitrary precision. Because every categorization method tries to establish a functional connection between group membership and object characteristics, pinpointing the underlying function is crucial. Third, because neural networks are nonlinear models, they may be used to simulate complicated real-world interactions. Finally, neural networks can estimate posterior probabilities, which is the foundation for developing classification rules and doing statistical analysis [42].
An example of Regression Loss Function for neural network is Mean Squared Error

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_1 - \hat{y}_i)^2$$

i is index of sample, $\hat{y}$ is predicted value, y is expected value, and N is number of

samples in dataset.

Although classification costs are difficult to assess in real-world issues, disregarding the uneven risk of misclassification for various groups might have a substantial influence on the classification's practical application. It should be noted that a neural classifier that reduces the overall number of misclassification mistakes may be ineffective in cases when the effects or costs of distinct misclassification errors are very unequal. Designing efficient cost-based neural network classifiers requires more research [42].

5.1.7 Decision Tree

This classification algorithm is a simple representation for classifying examples. It is a Supervised Machine Learning technique where the data is continuously split according to a certain parameter.
Roots, branches, and leaves make up a typical tree. Decision Tree follows the same structure. There are root nodes, branches, and leaf nodes in it. Every internal node tests an attribute, the result of the test is on the branch, and the class label as a consequence is on the leaf node. A root node is the topmost node in a Tree and serves as the parent of all nodes. A decision tree is a tree in which each node (attribute) represents a feature, each link (branch) represents a decision (rule), and each leaf represents a result (categorical or continuous value) [43].
A simple squared error can be used as cost function for a regression tree as shown below

$$E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

or Mean Squared Error

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Because decision trees are designed to mirror human reasoning, grabbing facts and making appropriate interpretations is a breeze. The goal is to build a tree like this for all of the data and process a single result at each leaf.
One of the finest features of Decision Tree is its transparency. Another benefit is the option to choose the most biased characteristic and the type of comprehensibility. It's also simple to categorize and interpret, and it works with both continuous and

discrete data sets. In terms of performance, non-linear has no effect on any of the decision tree's characteristics [43].

5.1.8 Random Forest

This is a tree-based and ensemble learning algorithm also used for classification and other tasks that operate by constructing a lot of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the object.
For a multiclass classification, separate loss for each class label for each observation is calculated by summing the result as below

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

M is the number of classes (UDP, TCP, ICMP and Normal), y is the binary indicator, and p is the predicted probability observation.

It is a set of tree predictors in which the values of a random vector selected separately and with the same distribution for all trees in the forest are used to forecast the behavior of each tree. As the number of trees in a forest grows larger, the generalization error converges to a limit. The strength of individual trees in the forest and their connection determines the generalization error of a forest of tree classifiers. When each node is divided using a random selection of features, the error rates are comparable to Adaboost [44].

5.1.9 Logistic Regression

This classification algorithm is used when the value of the target variable is categorical in nature. Logistic regression is commonly used when the data at hand has binary output, and it belongs to one class or another or is either a 0 or 1.
Various classification issues, such as spam detection, may be solved using logistic regression. Diabetes forecasting, determining if a consumer will buy a specific product or switch to a rival, determining whether a user will click on a specific marketing link, and many more scenarios are just a few examples. Logistic regression was adopted in this research paper to determine if packet is infected with ICMP, UPD, TCP flood attack or normal.
Cross-Entropy (Log Loss) as shown below is used for logistic regression cost function

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y_i log\sigma(mx+b) + (1-y_i)log(1-\sigma(mx+b))]$$

The principles of multiple regression analysis are extended to study scenarios where the outcome variable is categorical using logistic regression analysis (LRA). Situations with categorical outcomes are extremely prevalent in practice. The result variable, Y, is assumed to be categorical in the logistic regression analysis model, although LRA does not explicitly represent this outcome variable. LRA, on the other hand, is based on the probabilities associated with Y values [45].

5.1.10 Multinomial Naïve Bayes

This algorithm considers a feature vector where a given term represents the number of times it appears or very often i.e., frequency. It is suitable for classification with discrete features. This is type of classifier uses multinomially distributed data, such as the kind seen in text classification. Because it is quick and simple to construct, it is frequently used as a baseline in text categorization. Furthermore, with the right preprocessing, it may compete with more sophisticated algorithms like SVMs [46].

$$Pr(j) = log\pi_j + \sum_{i=1}^{|v|} log(1+f_i)log(Pr(i|j))$$

This method was used as one of the classifiers in this study to reach the level of prediction. The Bayes theorem-based statistical classification algorithms known as Nave Bayes algorithms assist discover the conditional likelihood of two occurrences (DDoS infected or normal) occurring based on the probabilities of each individual event occurring.

5.2 Semi-supervised Learning Algorithms

This form of machine learning sits midway between supervised and unsupervised learning, with supervised learning including a labeled dataset and unsupervised learning involving an unlabeled dataset. During the training phase, semi-supervised learning uses a labeled dataset for a big quantity of data and an unlabeled dataset for a small amount of data. Fig 5.1 depicts how semi-supervised learning was applied on the DDoS dataset.

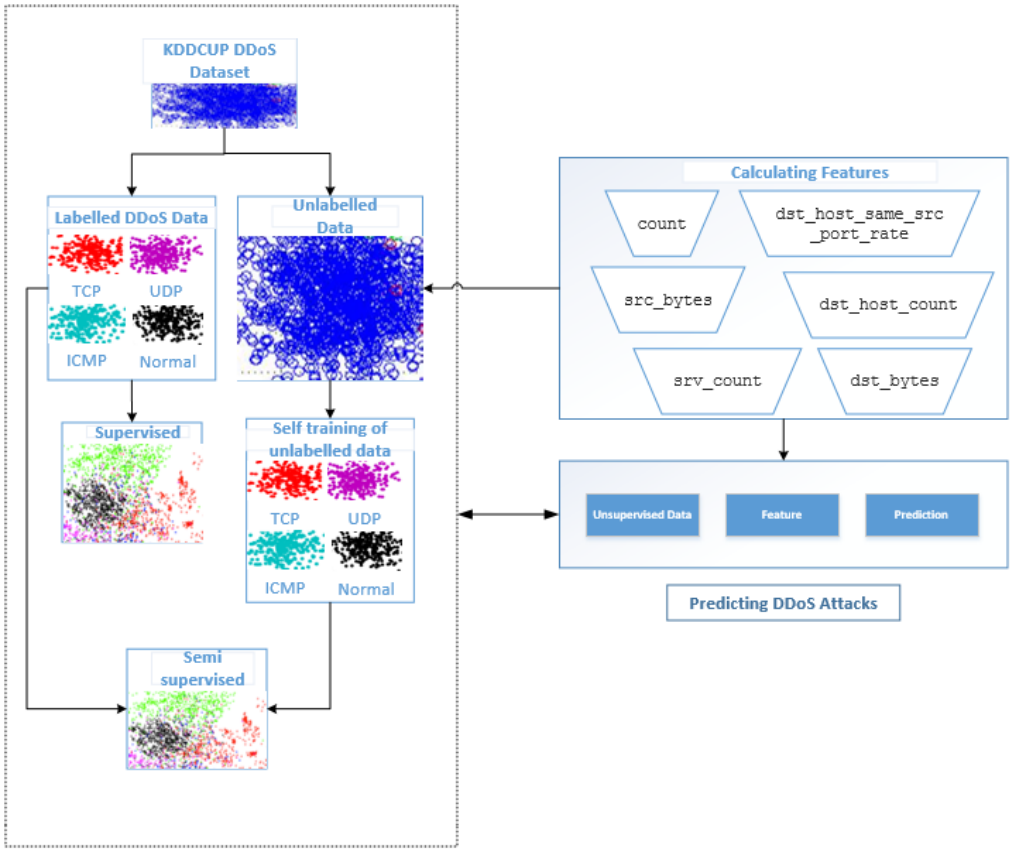

Figure 5.1: Architecture of Semi-supervised Learning

5.3 Deep Learning Algorithms

Deep Learning (DL) algorithm is a form of machine learning that mimics the human brain's data processing and pattern-making processes to aid decision-making. Deep Learning employs a multi-layered framework of algorithms capable of learning unsupervised from unstructured or unlabeled data. Deep neural learning or deep neural network (DNN) are other terms for the same phenomenon. This method can also detect DDoS attacks.

The goal of neural network approaches is to enable machines to see and perceive the world in the same way that humans do, and to use that knowledge for a variety of tasks such eg. classification. Advancements in Computer Vision using Deep Learning have been built and improved through time, largely through the use of this single algorithm.

DNN is a deep learning system that can take an image as input, assign significance to distinct items in the image, and distinguish between them. When compared to other classification algorithms, the amount of pre-processing required by a DNN is signif-

icantly less. While filters are hand-engineered in basic approaches, DNN can learn these filters/characteristics with adequate training. In addition, DNN is computationally efficient. It performs many mathematical calculations also known as layers. DNN models can now operate on any device, making them globally appealing.
The main building block of DNN is the hidden layer. DNNs are feedforward networks that transmit data from the input layer to the output layer without looping back. The DNN starts by creating a map of virtual neurons and assigning random integer values, or "weights" to their connections. This allows us to minimize the number of parameters, which reduces training time while also preventing overfitting.
A DNN architecture is utilized in this research to see if DL is an effective and early detection approach for DDoS attacks.

5.4 SMAC

Sequence model-based algorithm configuration (SMAC) is a powerful tool for fine-tuning algorithm parameters. It was recently discovered that it is extremely successful for hyperparameter optimization of machine learning algorithms, and that it scales better to large dimensions and discrete input dimensions than other techniques. Finally, the predictive models on which SMAC is built may collect and utilize key information about the model domain, such as the most significant input variables [47]. This hyper parameter optimization method was applied on the algorithm used for the DDoS detection.

Chapter 6

Experimental Results

Experiment using the seven feature selection techniques and ten different Machine learning classifiers has been carried out to evaluate the prediction accuracy of each algorithm, and to compare the results with semi-supervised and Deep learning techniques while using RadViz and Parallel Coordinates to visualize the features selected.

6.1 Data Description

Because finding an appropriate dataset for a certain sort of DDoS attack is difficult, most researchers either use existing datasets or develop their own. It is well known that DDoS attacks adapt innovative tactics and become more complex in order to overcome any security measures, implying that relying on a single dataset in a real-world scenario is not a smart idea. This is the reason several classifiers and feature selection techniques were used in this research work.
KDD Cup 1999 Data was the dataset used throughout this research work. It was the data set for the Third International Knowledge Discovery and Data Mining Tools Competition, which took place in connection with KDD-99, the Fifth International Conference on Knowledge Discovery and Data Mining. The goal of the competition was to create a network intrusion detector, which was a prediction model capable of differentiating between bad connections, often known as intrusions, and good regular connections. This database offers a standard set of auditable data, including a wide range of intrusions simulated in a military network environment [48].

The KDD training dataset comprises around 4,900,000 single connection vectors, each of which has 41 characteristics and is classified as either normal or attack, with only one attack type (DDoS, U2R, R2L and Probing attack). The DDoS attack is the subject of this study. The data set features can be classified into three groups [49]:

Basic Features: All of the attributes that may be retrieved from a TCP/IP connection are included in this category. The majority of these characteristics result in an implicit detection delay. There are nine features in this category as shown in Table 6.1

Table 6.1: Basic feature names and descriptions

Feature Name	Description
duration	length (number of seconds) of the connection
protocol_type	type of the protocol, e.g. tcp, udp, etc.
service	network service on the destination, e.g., http, telnet, etc.
src_bytes	number of data bytes from source to destination
dst_bytes	number of data bytes from destination to source
flag	normal or error status of the connection
land	1 if connection is from/to the same host/port; 0 otherwise
wrong_fragment	number of "wrong" fragments
urgent	number of urgent packets

Traffic features: Also called time-based features, this category contains characteristics computed with regard to a window interval, such as "same host" or "same service" features. Same Host features look at just the connections that have the same destination host as the current connection in the last two seconds and compute statistics on protocol behavior or service. While Same Service features look at only the connections that have had the same service as the current connection in the last two seconds. There are nine features in this category as shown in Table 6.2

Table 6.2: Traffic feature names and descriptions

Feature Name	Description
count	number of connections to the same host as the current connection in the past two seconds
serror_rate	% of connections that have "SYN" errors
rerror_rate	% of connections that have "REJ" errors
same_srv_rate	% of connections to the same service
diff_srv_rate	% of connections to different services
srv_count	number of connections to the same service as the current connection in the past two seconds
srv_serror_rate	% of connections that have "SYN" errors
srv_rerror_rate	% of connections that have "REJ" errors
srv_diff_host_rate	% of connections to different hosts

Content features: These features are content-based rather than frequency-based. To identify such attacks, some features are required, such as the ability to search for abnormal activity in the data portion, such as the number of failed login attempts. There are thirteen features in this category as shown in Table 6.3.

Table 6.3: Content feature names and descriptions

Feature Name	Description
hot	number of "hot" indicators
num_failed_logins	number of failed login attempts
logged_in	1 if successfully logged in; 0 otherwise
num_compromised	number of "compromised" conditions
root_shell	1 if root shell is obtained; 0 otherwise
su_attempted	1 if "su root" command attempted; 0 otherwise
num_root	number of "root" accesses
num_file_creations	number of file creation operations
num_shells	number of shell prompts
num_access_files	number of operations on access control files
num_outbound_cmds	number of outbound commands in an ftp session
is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise
is_guest_login	1 if the login is a "guest" login; 0 otherwise

6.2 Exploratory Data Analysis

Before any work on the data set could begin, it was necessary to have a high-level understanding of the data. Exploratory Data Analysis (EDA) aided in the analysis of my dataset and the visual summarization of the major characteristics.
This project work is based on DDoS attack prediction. There were three types of assaults in the dataset: TCP, UDP, and ICMP. There are also regular traffic data that is not affected by any of the three types of assaults. The frequency bar plot is given in Fig 6.1. The plot indicates that there are more ICMP attacks than other type of attacks in the dataset.

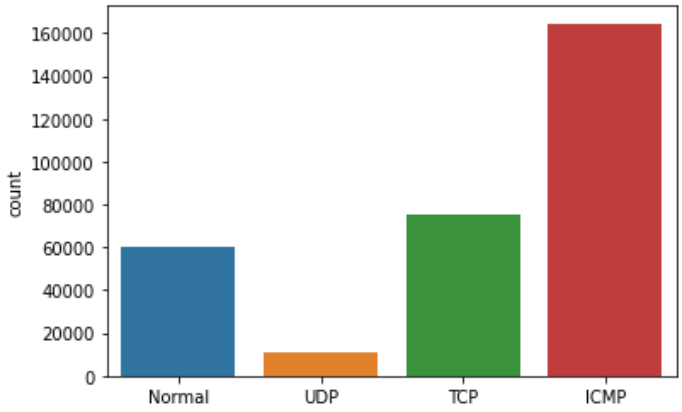

Figure 6.1: Bar plot showing frequency distribution of the dataset

The heat map in Fig. 6.2 illustrates the feature-to-feature connection, which helps to visualize the dataset's attributes. It depicts the magnitude of the relationship between the features in the dataset. The clustered fourth quadrant denotes a strong connection.

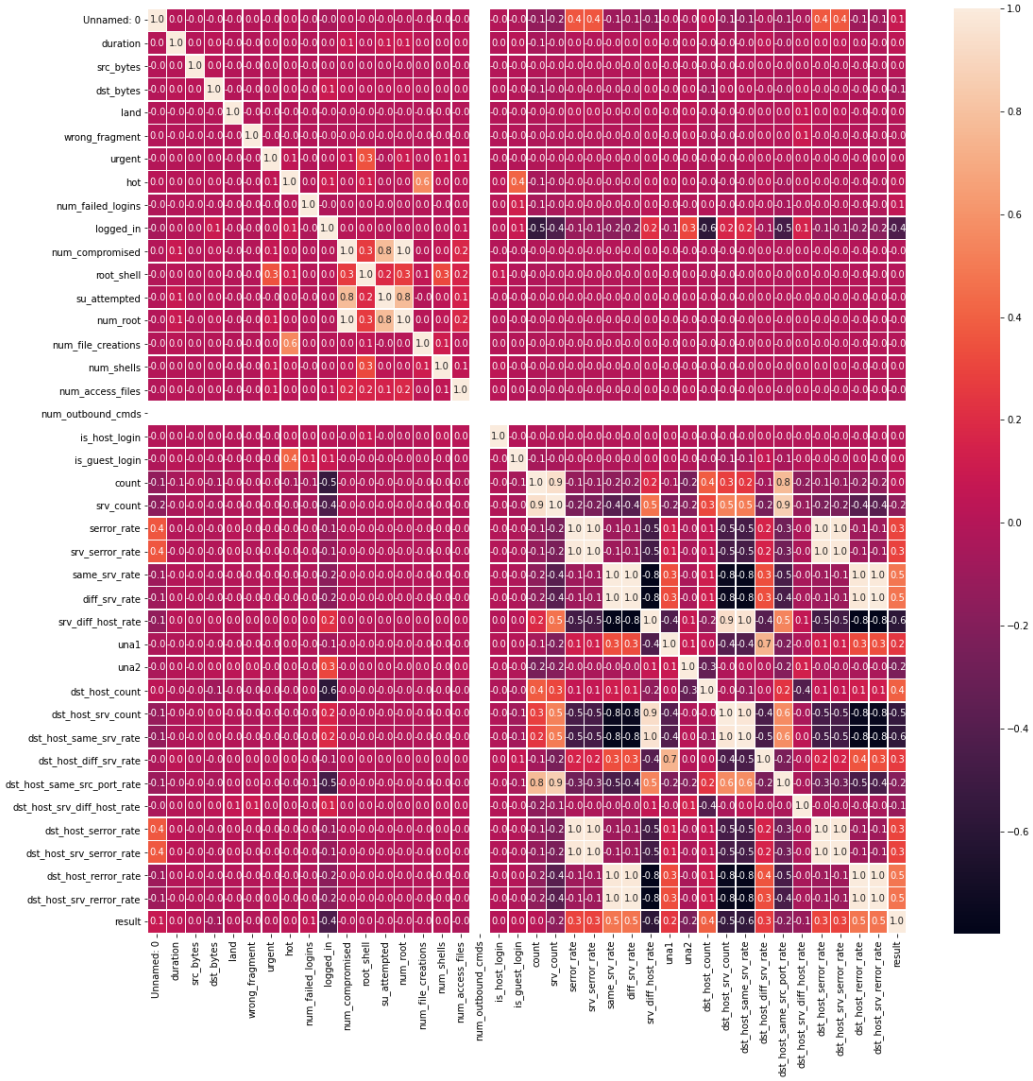

Figure 6.2: Heat map showing feature to feature relationship

The three distinct assaults, as well as regular traffic, were evaluated using the principal component analysis (PCA) depicted in Fig 6.3. The PCA assisted in reducing the dataset's dimensionality while maintaining the information.

The dataset utilized in this study was additionally analyzed and explored using an unsupervised, non-linear approach called t-Distributed Stochastic Neighbor Embedding (t-SNE). It offers an impression of how the data was organized in a multidimensional space. The t-SNE is seen in Figure 6.4.

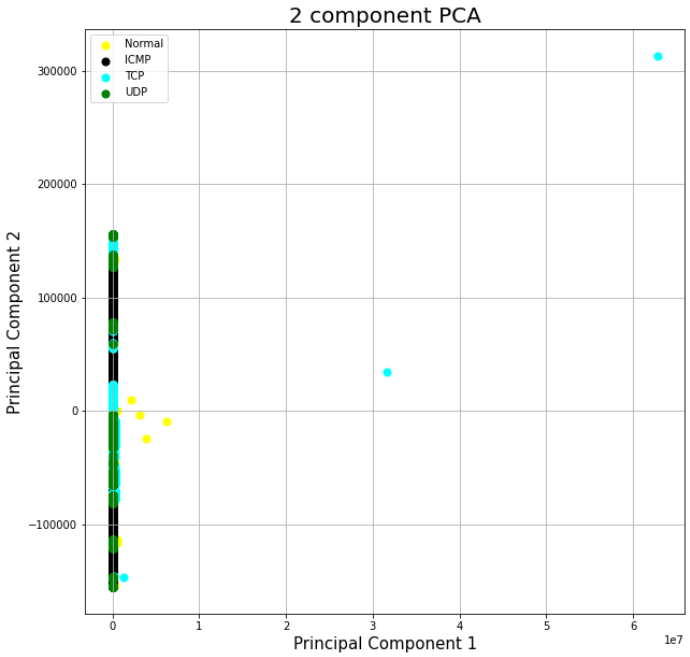

Figure 6.3: Principal Component Analysis

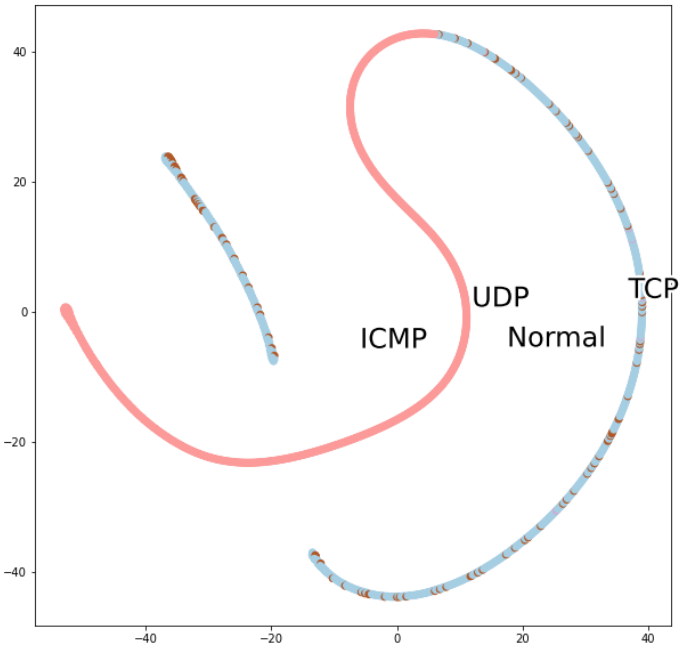

Figure 6.4: t-Distributed Stochastic Neighbor Embedding

6.3 Feature Selection Visualization

There is need to understand the dataset by selecting the best features and visualizing them.

6.3.1 Feature Selection Results

This research used the three selection methods, Filter, Wrapper and Embedded. The following tables list the features that filter method has selected as the most relevant in the prediction of the three types of DDoS volumetric assault – TCP, UDP and ICMP flood attack.

Variance Threshold: This technique of Filter method has selected the following features in Table 6.4 as important.

Table 6.4: Selected Features of Variance Threshod

duration	count	srv_diff_host_rate
src_bytes	srv_count	dst_host_count
dst_bytes	serror_rate	dst_host_srv_count
logged_in	srv_serror_rate	dst_host_same_srv_rate
num_compromised	same_srv_rate	dst_host_same_src_port_rate
num_root	diff_srv_rate	dst_host_serror_rate
dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate

Mutual Information: This second filter method has selected just four features as illustrated in the bar graph of Fig 6.5

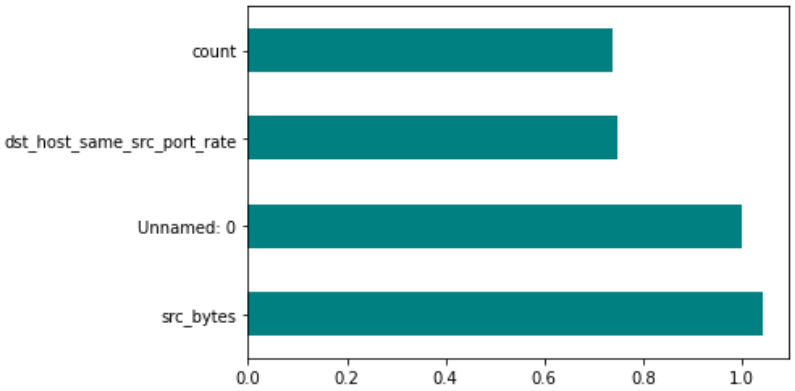

Figure 6.5: Selected Features of Mutual Information

ANOVA F-value: This filter-based method has selected four features that are relevant to the prediction of DDoS as shown in Fig 6.6

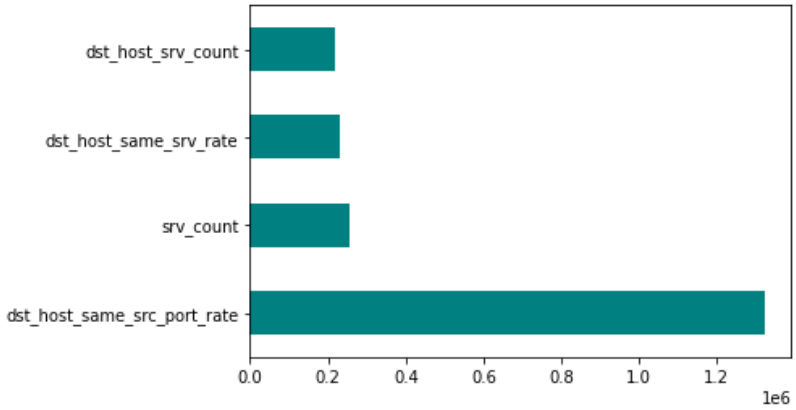

Figure 6.6: Selected Features of ANOVA F-Value

Selectk Best: This is another filter-based method that also has five features (src_bytes, dst_bytes, count, Unnamed: 0, and srv_count) relevant to the prediction of DDoS attack as shown in Fig 6.7

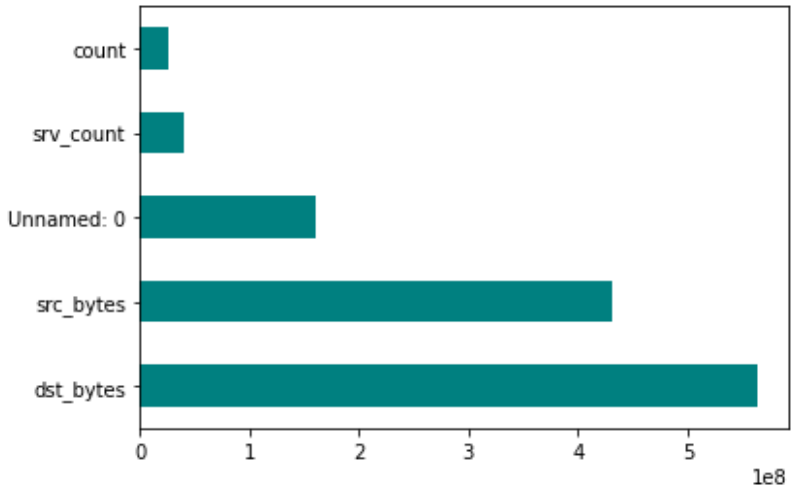

Figure 6.7: Selected Features of SelectK Best

Pearson Correlation: This is the fifth technique of filter method used in this research to predict DDoS attack. It has selected seven attributes as shown in Table 6.5

Table 6.5: Selected Features of Pearson Correlation

dst_bytes	same_srv_rate
num_failed_logins	dst_host_count
logged_in	dst_host_srv_diff_host_rate
serror_rate	

RFE Wrapper: The RFE technique of Wrapper method used in this research to

predict DDoS attack has the weight of attributes as shown in Fig 6.8

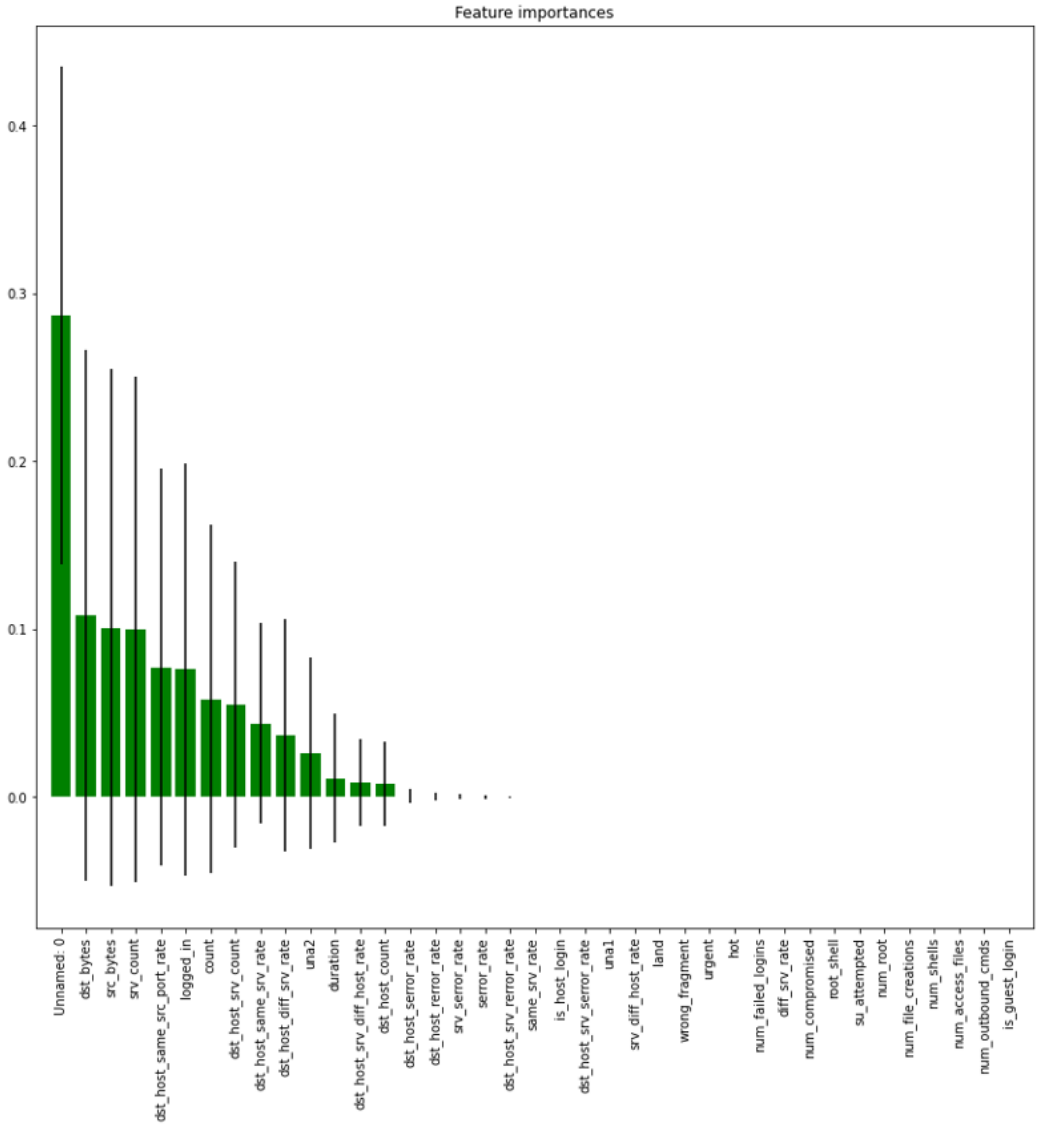

Figure 6.8: Weights of RFE Wrapper Features

6.3.2 Visualization Results

Some feature selection technique used in this research work was visualized using RadViz and Parallel Coordinate.

RadViz is best used with data manipulation with low cluttering. RadViz was used to visualize Mutual Information, ANOVA F-value, and SelectK Best technique, because these techniques have chosen a small number of features (low clutter) as the relevant attributes for prediction of DDoS traffic. The RadViz for Mutual information, ANOVA F-value and SelectK Best are shown in Fig 6.9, 6.10 and 6.11 respectively.

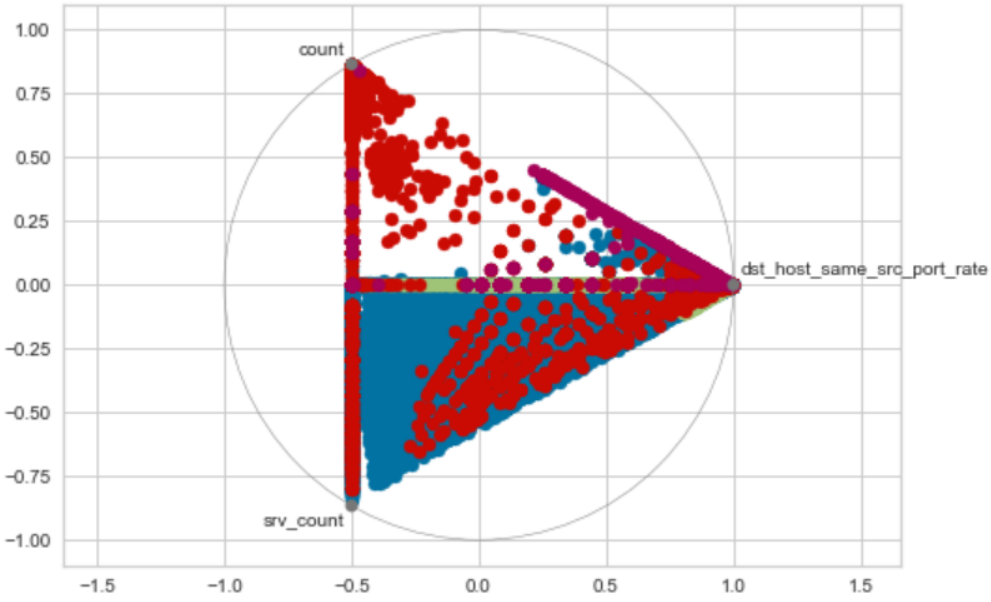

Figure 6.9: RadViz visualization of mutual information features

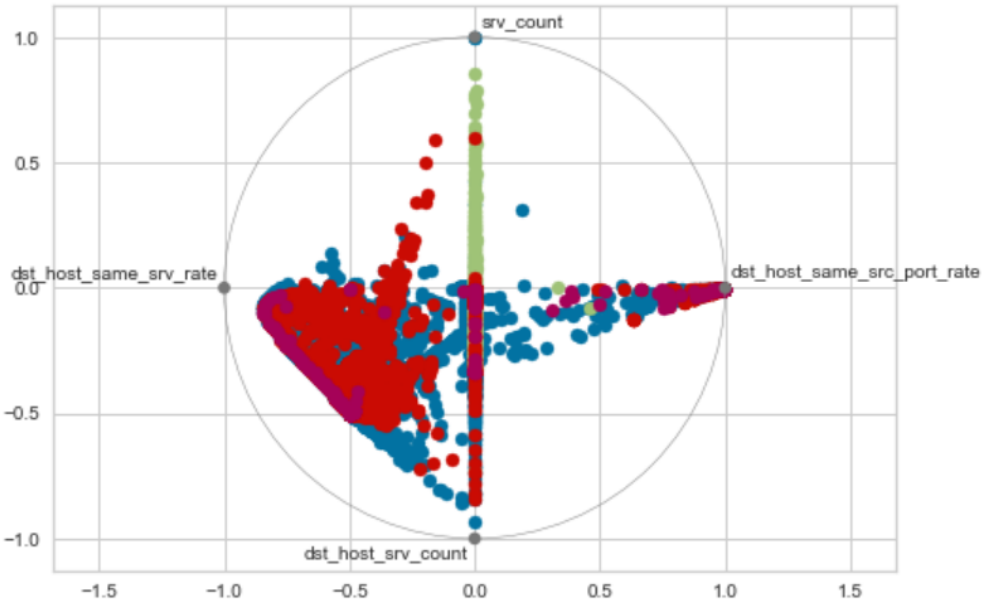

Figure 6.10: RadViz visualization of ANOVA F-value features

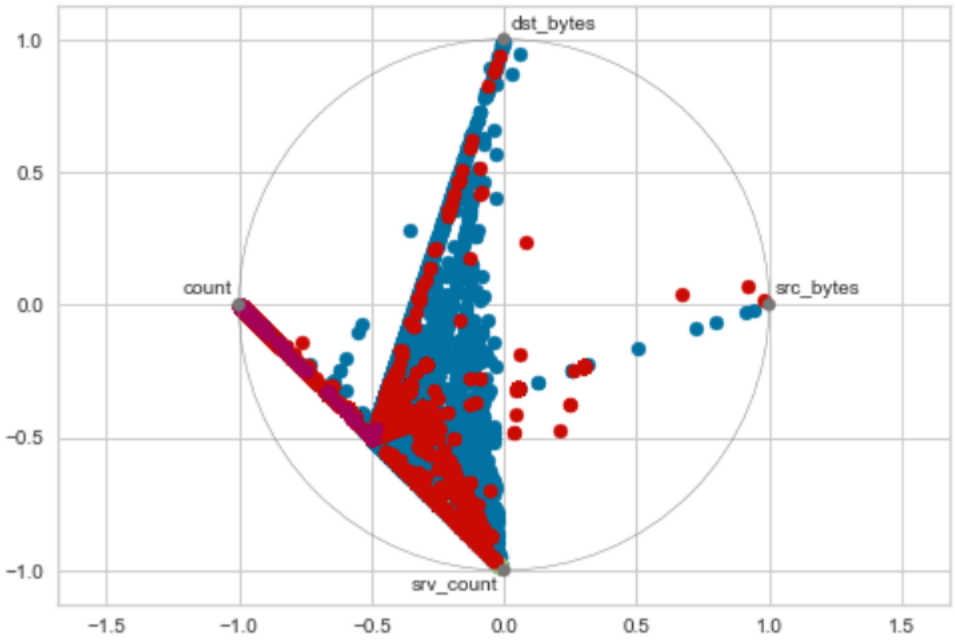

Figure 6.11: RadViz visualization of SelectK Best features

While the parallel coordinate plots for Mutual information, ANOVA F-value and SelectK Best are shown in Fig 6.12, 6.13 and 6.14 respectively

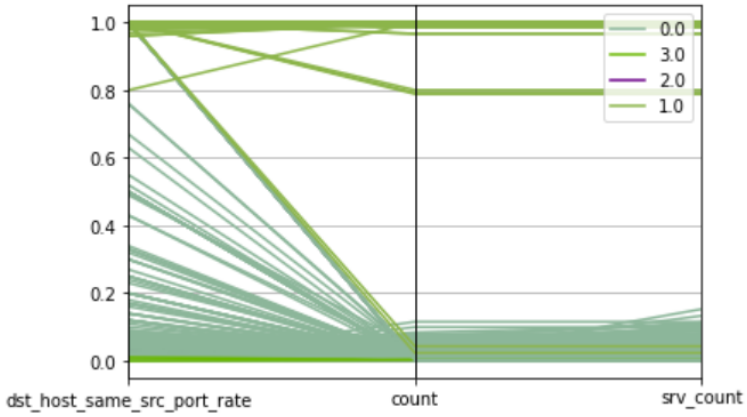

Figure 6.12: Parallel Coordinate Plot for Mutual Information Features

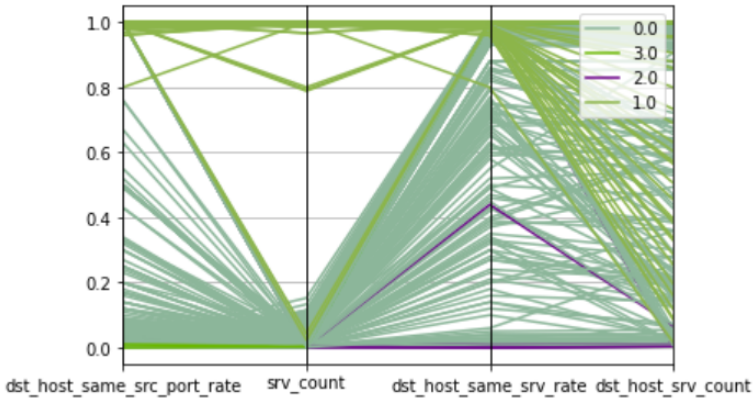

Figure 6.13: Parallel Coordinate Plot for ANOVAF Value Features

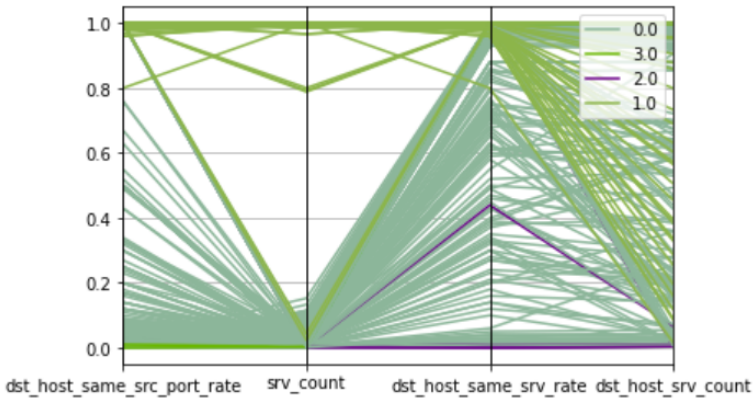

Figure 6.14: Parallel Coordinate Plot for Selectk Best Features

6.4 Result of ML-based algorithms

The 10 different classifiers generated varied outputs under the different feature selection methods. The outcomes are shown in the Tables 6.6 to 6.16
Random forest classifier produced the best result under Variance threshold selection technique as highlighted in Table 6.6

Table 6.6: Classifier results under Variance Threshold

	Classifier	**train_score**	**test_score**	**train_time**
0	Gradient Boosting Classifier	0.984622	0.984846	156.194912
1	**Random Forest**	**1.000000**	**0.990376**	13.561750
2	Logistic Regression	0.881283	0.881105	11.809474
3	Nearest Neighbors	0.987484	0.981084	32.719991
4	Decision Tree	1.000000	0.989679	0.740518
5	Linear SVM	0.825299	0.828044	3810.702263
6	Neural Net	0.947166	0.947797	88.413321
7	Naive Bayes	0.893372	0.894587	0.180190
8	Multinomial Naive Baes	0.675253	0.673290	0.085534
9	AdaBoost	0.782386	0.784437	10.133502

Under the mutual information selection approach, the decision tree classifier generated the best train results, whereas random forest produced the best test results, as shown in Table 6.7

Table 6.7: Classifier results under Mutual Information

	Classifier	**train_score**	**test_score**	**train_time**
0	Gradient Boosting Classifier	0.967279	0.966595	84.083031
1	**Random Forest**	0.999986	**0.974290**	16.942631
2	Logistic Regression	0.901415	0.902367	8.082144
3	Nearest Neighbors	0.968124	0.953359	0.907448
4	**Decision Tree**	**1.000000**	0.973882	0.325840
5	Linear SVM	0.805907	0.808636	1610.936481
6	Neural Net	0.890263	0.890932	145.242915
7	Naive Bayes	0.737268	0.740014	0.050088
8	Multinomial Naive Bayes	0.723769	0.725611	0.054588
9	AdaBoost	0.911175	0.909548	6.426021

Decision tree classifier also produced the best train result and random forest produced the best test result under anova selection technique as highlighted in Table 6.8

Table 6.8: Classifier results under ANOVA F-value

	Classifier	**train_score**	**test_score**	**train_time**
0	Gradient Boosting Classifier	0.961359	0.960025	81.518493
1	**Random Forest**	0.999954	**0.964505**	14.478940
2	Logistic Regression	0.901415	0.902367	8.118055
3	Nearest Neighbors	0.968055	0.953316	0.756841
4	**Decision Tree**	**1.000000**	0.964355	0.409253
5	Linear SVM	0.806196	0.809075	1203.156768
6	Neural Net	0.858346	0.860228	142.749018
7	Naive Bayes	0.736735	0.739479	0.044386
8	Multinomial Naive Bayes	0.723682	0.725557	0.059081
9	AdaBoost	0.918207	0.918647	6.021598

Under the anova selection approach, the decision tree and random forest classifier

generated very good accuracy results, as shown in Table 6.9

Table 6.9: Classifier results under Selectk Best

	Classifier	**train_score**	**test_score**	**train_time**
0	Gradient Boosting Classifier	0.983580	0.983249	92.191409
1	**Random Forest**	**1.000000**	**0.990355**	12.956010
2	Logistic Regression	0.832147	0.833778	9.759601
3	Nearest Neighbors	0.986446	0.979445	20.981298
4	**Decision Tree**	**1.000000**	0.989969	0.355325
5	Linear SVM	0.827329	0.830070	1899.876206
6	Neural Net	0.857657	0.856348	154.133546
7	Naive Bayes	0.654079	0.653720	0.071598
8	Multinomial Naive Bayes	0.673066	0.671264	0.059828
9	AdaBoost	0.774091	0.774384	6.662487

Decision tree classifier also produced the best train result and random forest produced the best test result under pearson correlation selection technique as highlighted in Table 6.10

Table 6.10: Classifier results under Pearson Correlation

	Classifier	**train_score**	**test_score**	**train_time**
0	Gradient Boosting Classifier	0.976741	0.976669	91.087686
1	**Random Forest**	0.999977	**0.981942**	21.408278
2	Logistic Regression	0.662502	0.661383	8.463850
3	Nearest Neighbors	0.984448	0.977108	75.237094
4	**Decision Tree**	**1.000000**	0.981695	0.564909
5	Linear SVM	0.820007	0.823104	2842.958874
6	Neural Net	0.699600	0.697050	153.488644
7	Naive Bayes	0.661184	0.660247	0.091063
8	Multinomial Naive Bayes	0.642252	0.641771	0.070393
9	AdaBoost	0.821955	0.819289	6.890245

Table 6.11 shows that using the wrapper selection approach, the decision tree and random forest classifier generated very high accuracy results.

Table 6.11: Classifier results under RFE Wrapper Method

	Classifier	**train_score**	**test_score**	**train_time**
0	Gradient Boosting Classifier	0.971822	0.972253	69.722547
1	**Random Forest**	**0.975583**	**0.973904**	9.894682
2	Logistic Regression	0.747460	0.745973	12.081149
3	Nearest Neighbors	0.971105	0.969553	15.989417
4	**Decision Tree**	**0.975583**	0.973239	0.255505
5	Linear SVM	0.791347	0.793000	2144.402646
6	Neural Net	0.925845	0.926234	45.681469
7	Naive Bayes	0.829474	0.830102	0.135770
8	Multinomial Naive Bayes	0.579671	0.578819	0.136557
9	AdaBoost	0.726736	0.724314	11.129388

Under the embedded selection approach, both the decision tree and the random

forest classifier achieved very high accuracy results, as shown in Table 6.12.

Table 6.12: Classifier results under Embedded Method

	Classifier	**train_score**	**test_score**	**train_time**
0	Gradient Boosting Classifier	0.977756	0.978062	147.495746
1	**Random Forest**	**0.980498**	**0.980141**	18.521789
2	Logistic Regression	0.908148	0.907329	22.901584
3	Nearest Neighbors	0.976125	0.975758	71.816976
4	**Decision Tree**	**0.980498**	0.979766	0.632456
5	Linear SVM	0.834443	0.836940	3368.045583
6	Neural Net	0.957331	0.957743	76.405041
7	Naive Bayes	0.893910	0.895176	0.149778
8	Multinomial Naive Bayes	0.691870	0.689194	0.150126
9	AdaBoost	0.915878	0.916535	12.453486

6.5 Result of Semi-supervised Learning

Without using any feature selection approaches, the ten classifiers were utilized to categorize DDoS attacks. Table 6.13 shows the accuracy, precision, recall, and Kappa values for each classifier. Decision tree has produced the highest accuracy, recall, F1 and kapper scores while linear discriminant produced the best precision.

Table 6.13: Supervised Classification

	Classifier	**Accuracy**	**Precision**	**Recall**	**F1**	**Kappa**
0	Nearest Neighbors	0.953	0.949824	0.953	0.950912	0.909701
1	LogisticRegression	0.930	0.913734	0.930	0.919363	0.863928
2	**Decision Tree**	**0.958**	0.956903	**0.958**	**0.957262**	**0.919624**
3	Random Forest	0.940	0.889845	0.940	0.912887	0.881547
4	MLP	0.939	0.921932	0.939	0.922573	0.880436
5	AdaBoost	0.942	0.895254	0.942	0.916665	0.885671
6	GaussianNB	0.950	0.961780	0.950	0.951926	0.906560
7	**LinearDiscriminant**	0.934	**0.961185**	0.934	0.940918	0.878062
8	GradientBoost	0.942	0.937503	0.942	0.939238	0.888504
9	MultinomialNB	0.834	0.946111	0.834	0.853962	0.707867

The outcome of the semi-supervised classification, which used both labeled and unlabeled data to predict DDoS, is given in Table 6.14. Decision tree has produced the best scores for accuracy, recall, F1 and kapper.

Table 6.14: Semi-supervised classification

	Classifier	Accuracy	Precision	Recall	F1	Kappa
0	Nearest Neighbors	0.957576	0.947576	0.957576	0.952059	0.918841
1	LogisticRegression	0.927273	0.879647	0.927273	0.901414	0.857958
2	**Decision Tree**	**0.963636**	**0.954091**	**0.963636**	**0.958317**	**0.930435**
3	Random Forest	0.936364	0.882626	0.936364	0.907229	0.875101
4	MLP	0.945455	0.938636	0.945455	0.933396	0.894253
5	AdaBoost	0.936364	0.882626	0.936364	0.907229	0.875101
6	GaussianNB	0.927273	0.945627	0.927273	0.930278	0.866214
7	LinearDiscriminant	0.933333	0.949182	0.933333	0.936635	0.877026
8	GradientBoost	0.960606	0.950924	0.960606	0.953778	0.924254
9	MultinomialNB	0.824242	0.924924	0.824242	0.840580	0.691176

When supervised classification was compared to semi-supervised classification, it was discovered that the supervised method performed somewhat better. The comparison is shown in Table 6.15.

Table 6.15: Comparison between supervised and Semi-supervised classification

	Classifier	Supervised train score	Supervised test score	Semi-supervised train score	Semi-supervised test score
0	Nearest Neighbors	0.977	0.953	0.967665	0.957576
1	LogisticRegression	0.935	0.930	0.933533	0.927273
2	Decision Tree	0.966	0.958	0.961677	0.963636
3	Random Forest	0.948	0.940	0.946707	0.936364
4	MLP	0.959	0.939	0.949102	0.945455
5	AdaBoost	0.951	0.942	0.948503	0.936364
6	GaussianNB	0.957	0.950	0.909581	0.927273
7	LinearDiscriminant	0.936	0.934	0.916766	0.933333
8	GradientBoost	0.996	0.942	0.971257	0.960606
9	MultinomialNB	0.823	0.834	0.799401	0.824242

6.6 Result of DL Algorithm

The Deep Learning model produced a training accuracy of 85.50 percent and validation accuracy of 86.67 percent. The accuracy is somewhat low compared to the best result of the supervised and semi-supervised-based models. The result is displayed in Table 6.16

Table 6.16: Deep learning results

Algorithm	Training accuracy	Test accuracy	Precision	Recall	F1
Deep Learning	0.8550	0.8667	0.87	0.87	0.87

Chapter 7

Conclusion and Future Works

This study only used a single dataset since machine learning approaches rely on the training phase to learn from a given dataset and develop a learning profile to find patterns, restricting the applicability of the methods used. It used ten classifiers under different feature selection techniques and conditions.

Based on the findings, it can be deduced that some classifiers consistently perform well (above 95 percent) in all types of supervised and semi-supervised learning (with and without feature selection approaches). Random Forest, Nearest Neighbors, and Decision Tree are some of them. When compared to other methods, Multinomial Naive Bayes has demonstrated low prediction ability (less than 85 percent) on both supervised and semi-supervised models. When accuracy is a priority, machine learning techniques such as Random Forest, Nearest Neighbors, and Decision Tree are recommended for DDoS categorization. Similarly, some classifiers take a long time to run, while others take only a few seconds. The slowest method was SVM, while the quickest methods were Naive Bayes and Multinomial Naive Bayes.

DNN-based and Semi-supervised learning also produced reasonable good accuracies of graeter than 85 percent for DNN and an average of 92 percent for both train and test score of semi-supervised model.

Bibliography

[1] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms a classification," in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology IEEE Cat No 03EX795*, Jan. 2003, pp. 190–193. DOI: 10.1109/ISSPIT.2003.1341092.

[2] A. Aljuhani, "Machine learning approaches for combating distributed denial of service attacks in modern networking environments," *IEEE Access*, vol. 9, pp. 42 236–42 264, 2021. DOI: 10.1109ACCESS.2021.3062909.

[3] P. Khuphiran, P. Leelaprute, P. Uthayopas, K. Ichikawa, and W. Watanakeesuntorn, "Performance comparison of machine learning models for ddos attacks detection," in *2018 22nd International Computer Science and Engineering Conference ICSEC*, 2018, pp. 1–4. DOI: 10.1109/ICSEC.2018.8712757.

[4] J. N. Bakker, N. Bryan, and K. G. S. Winston, "Can machine learning techniques be effectively used in real networks against ddos attacks?" In *2018 27th International Conference on Computer Communication and Networks ICCCN*, 2018, pp. 1–6. DOI: 10.1109/ICCCN.2018.8487445.

[5] P. S. Singh, S. Behal, and S. Bhatia, "Detection of ddos attacks using machine learning algorithms," in *2020 7th International Conference on Computing for Sustainable Global Development INDIACom*, 2020, pp. 16–21. DOI: 10.23919/INDIACom49435.2020.9083716.

[6] S. P. S, M. Sivaram, D. Yuvaraj, and A. Jayanthiladevi, "Machine learning based ddos detection," in *2020 International Conference on Emerging Smart Computing and Informatics ESCI*, 2020, pp. 234–237. DOI: 10.1109/ESCI48226.2020.9167642.

[7] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops SPW*, 2018, pp. 29–35. DOI: 10.1109/SPW.2018.00013.

[8] M. H. Aysa, A. A. Ibrahim, and A. H. Mohammed, "Iot ddos attack detection using machine learning," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies ISMSIT*, 2020, pp. 1–7. DOI: 10.1109/ISMSIT50672.2020.9254703.

[9] G. Ajeetha and G. M. Priya, "Machine learning based ddos attack detection," in *2019 Innovations in Power and Advanced Computing Technologies i-PACT*, vol. 1, 2019, pp. 1–5. DOI: 10.1109/i-PACT44901.2019.8959961.

[10] M. Barati, A. Abdullah, N. I. Udzir, R. Mahmod, and N. Mustapha, "Distributed denial of service detection using hybrid machine learning technique," in *2014 International Symposium on Biometrics and Security Technologies ISBAST*, 2014, pp. 268–273. DOI: 10.1109/ISBAST.2014.7013133.

[11] A. Sudugala, W. Chanuka, A. Eshan, U. Bandara, and K. Abeywardena, "Wanheda a machine learning based ddos detection system," in *2020 2nd International Conference on Advancements in Computing ICAC*, vol. 1, 2020, pp. 380–385. DOI: 10.1109/ICAC51239.2020.9357130.

[12] H. Zecheng, T. Zhang, and R. Lee, "Machine learning based ddos attack detection from source side in cloud," Jun. 2017, pp. 114–120. DOI: 10.1109/CSCloud.2017.58.

[13] T.-K. Luong, T.-D. Tran, and G.-T. Le, "Ddos attack detection and defense in sdn based on machine learning," in *2020 7th NAFOSTED Conference on Information and Computer Science NICS*, 2020, pp. 31–35. DOI: 10.1109/NICS51282.2020.9335867.

[14] B. K. Devi, G. Preetha, G. Selvaram, and S. M. Shalinie, "An impact analysis real time ddos attack detection and mitigation using machine learning," in *2014 International Conference on Recent Trends in Information Technology*, 2014, pp. 1–7. DOI: 10.1109/ICRTIT.2014.6996133.

[15] K. Wehbi, L. Hong, T. Al-salah, and A. Bhutta, "A survey on machine learning based detection on ddos attacks for iot systems," in *2019 SoutheastCon*, 2019, pp. 1–6. DOI: 10.1109/SoutheastCon42311.2019.9020468.

[16] M. Zekri, S. E. Kafhali, N. Aboutabit, and Y. Saadi, "Ddos attack detection using machine learning techniques in cloud computing environments," in *2017 3rd International Conference of Cloud Computing Technologies and Applications CloudTech)*, 2017, pp. 1–7. DOI: 10.1109/CloudTech.2017.8284731.

[17] B. Hussain, Q. Du, B. Sun, and Z. Han, "Deep learning-based ddos-attack detection for cyber–physical system over 5g network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 860–870, 2021. DOI: 10.1109/TII.2020.2974520.

[18] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincon, and D. Siracusa, "Lucid a practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020. DOI: 10.1109/TNSM.2020.2971776.

[19] L. Wang and Y. Liu, "A ddos attack detection method based on information entropy and deep learning in sdn," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference ITNEC*, vol. 1, 2020, pp. 1084–1088. DOI: 10.1109/ITNEC48623.2020.9085007.

[20] L. Ma, Y. Chai, L. Cui, D. Ma, Y. Fu, and A. Xiao, "A deep learning-based ddos detection framework for internet of things," in *ICC 2020 - 2020 IEEE International Conference on Communications ICC*, 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9148944.

[21] M. E. Said, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet a deep-learning model for detecting network attacks," in *2020 IEEE 21st International Symposium on A World of Wireless Mobile and Multimedia Networks WoWMoM*, 2020, pp. 391–396. DOI: 10.1109/WoWMoM49955.2020.00072.

[22] J. Wang, Y. Liu, W. Su, and H. Feng, "A ddos attack detection based on deep learning in software-defined internet of things," in *2020 IEEE 92nd Vehicular Technology Conference VTC2020-Fall*, 2020, pp. 1–5. DOI: 10.1109/VTC2020-Fall49728.2020.9348652.

[23] J. He, Y. Tan, W. Guo, and M. Xian, "A small sample ddos attack detection method based on deep transfer learning," in *2020 International Conference on Computer Communication and Network Security CCNS*, 2020, pp. 47–50. DOI: 10.1109/CCNS50731.2020.00019.

[24] A. Bhati, A. Bouras, U. A. Qidwai, and A. Belhi, "Deep learning based identification of ddos attacks in industrial application," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability WorldS4*, 2020, pp. 190–196. DOI: 10.1109/WorldS450073.2020.9210320.

[25] wireshark.org, *Wireshark - go deep*, 2021. [Online]. Available: https://www.wireshark.org.

[26] S. Roy, P. Sharma, K. Nath, D. Bhattacharyya, and J. Kalita, "Pre-processing a data preparation step," in Jan. 2018. DOI: 10.1016/B978-0-12-809633-8.20457-3.

[27] R. Dash, R. Paramguru, and R. Dash, "Comparative analysis of supervised and unsupervised discretization techniques," *Int. J. Adv. Sci. Technol.*, vol. 2, Apr. 2011.

[28] R. Kohavi and J. George, "Wrappers for feature selection," *Artificial Intelligence - AI*, vol. 1, Jan. 1997.

[29] S. Ferat, "A survey on feature selection methods," *Computers and Electrical Engineering*, vol. 40, pp. 16–28, Jan. 2014. DOI: 10.1016/j.compeleceng.2013.11.024.

[30] V. Luhaniwal, *Feature selection using wrapper method - python implementation*, Dec. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/.

[31] Y. Charfaoui, *Feature selection using wrapper method - python implementation*, Dec. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/.

[32] G. Borboudakis and I. Tsamardinos, "Forward-backward selection with early dropping," *CoRR*, vol. abs/1705.10770, 2017. arXiv: 1705.10770. [Online]. Available: http://arxiv.org/abs/1705.10770.

[33] A. JainAarshay, *A complete tutorial on ridge and lasso regression in python*, Oct. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/01/ridge-lasso-regression-python-complete-tutorial/.

[34] B. Krishnapuram, A. Harternink, L. Carin, and M. Figueiredo, "A bayesian approach to joint feature selection and classifier design," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1105–1111, 2004. DOI: 10.1109/TPAMI.2004.55.

[35] M. Angelini, G. Blasilli, S. Lenti, A. Palleschi, and G. Santucci, "Towards enhancing radviz analysis and interpretation," in *2019 IEEE Visualization Conference (VIS)*, 2019, pp. 226–230. DOI: 10.1109/VISUAL.2019.8933775.

[36] J. Johansson and C. Forsell, "Evaluation of parallel coordinates: Overview categorization and guidelines for future research," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, pp. 1–1, Nov. 2015. DOI: 10.1109/TVCG.2015.2466992.

[37] G. Gongde, W. Hui, B. David, B. Yaxin, and G. Kieran, "Knn model based approach in classification," vol. 2888, Jan. 2003, pp. 986–996. DOI: 10.1007/978-3-540-39964-3_62.

[38] N. Alexey and K. Alois, "Gradient boosting machines a tutorial," *Frontiers in Neurorobotics*, vol. 7, p. 21, 2013. DOI: 10.3389/fnbot.2013.00021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2013.00021.

[39] T. Chengsheng, L. Huachen, and X. Bing, "Adaboost typical algorithm and its application research," *MATEC Web of Conferences*, vol. 139, p. 00 222, Jan. 2017. DOI: 10.1051/matecconf/201713900222.

[40] T. Yingjie, S. Yong, and L. Xiaohui, "Recent advances on support vector machines research," *Technological and Economic Development of Economy*, vol. 18, Mar. 2012. DOI: 10.3846/20294913.2012.661205.

[41] R. Irina, "An empirical study of the naïve bayes classifier," *IJCAI 2001 Work Empir Methods Artif Intell*, vol. 3, Jan. 2001.

[42] Z. Peter, "Neural networks for classification a survey," *Systems Man and Cybernetics Part C Applications and Reviews IEEE Transactions on*, vol. 30, pp. 451–462, Dec. 2000. DOI: 10.1109/5326.897072.

[43] P. Harsh and P. Purvi, "Study and analysis of decision tree based classification algorithms," *International Journal of Computer Sciences and Engineering*, vol. 6, pp. 74–78, Oct. 2018. DOI: 10.26438/ijcse/v6i10.7478.

[44] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001. DOI: 10.1023/A:1010950718922.

[45] C. Dayton, "Logistic regression analysis," Jan. 1992.

[46] S. Xu, Y. Li, and W. Zheng, "Bayesian multinomial naïve bayes classifier to text classification," May 2017, pp. 347–352. DOI: 10.1007/978-981-10-5041-1_57.

[47] F. Hutter, H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*, C. C. A, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523.

[48] [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[49] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528.