Network Intrusion Detection for Distributed Denial-of-Service (DDoS) Attacks using Machine Learning Classification Techniques

by

Yasar Shahid Hussain
Honours Bachelor of Science (H.B.Sc.), University of Toronto, 2011

A Report Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

©Yasar Shahid Hussain, 2020
University of Victoria

All rights reserved. This report may not be reproduced in whole or in part, by photocopying or other means, without the permission of the author.

Network Intrusion Detection for Distributed Denial-of-Service (DDoS) Attacks using Machine Learning Classification Techniques

by

Yasar Shahid Hussain
Honours Bachelor of Science (H.B.Sc.), University of Toronto, 2011

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Distributed Denial-of-Service (DDoS) is the number one cyber threat to the availability of business networks, applications, and services [1]. DDoS is a malicious attempt to disrupt normal traffic to a target server, service or network by overwhelming the target with illegitimate traffic. The consequences can be devastating such as financial losses, loss of productivity, brand damage, credit and insurance rating downgrades, compromised customer and supplier relationships, and IT budget overruns [2]. DDoS attacks continue to rise in complexity, volume and frequency, threatening the network security of all enterprises, regardless of their size [1]. The number of DDoS attacks is predicted to almost double to 14.5 million in 2022 compared to 2017 [3]. In 2017, the top motivations behind these attacks were criminals demonstrating attack capabilities, gaming, and extortion [1].

There is a critical need to devise Network Intrusion Detection Systems (NIDSs) to accurately predict DDoS attacks. In this work, supervised Machine Learning (ML) techniques are evaluated using the CICDDoS2019 dataset which consist of 80 network traffic features with benign (legitimate) traffic and 12 DDoS attacks [4]. This dataset was modified to create six datasets with the 24 best features [4] to predict DDoS attacks and benign traffic by employing undersampling and oversampling techniques. The ML algorithms evaluated are Bayesian Network (BayesNet), Bootstrap Aggregating (Bagging), k-Nearest Neighbors (kNN), Sequential Minimal Optimization (SMO), and Simple Logistic. The Waikato Environment for Knowledge Analysis (WEKA) tool is used for implementing the ML algorithms using k-fold ($k = 5$) cross validation. The evaluation metrics precision, recall, F-measure, True Positive Rate (TPR), False Positive Rate (FPR), and execution time are determined for the six datasets. The results obtained show that Bagging provides the best overall performance followed by kNN, BayesNet,

SMO, and Simple Logistic. Further, the execution time is approximately linear with the dataset size.

Contents

List of Tables

List of Figures

Abbreviations

AI Artificial Intelligence

API Application Programming Interface

DDoS Distributed Denial-of-Service

DNS Domain Name System

GUI Graphical User Interface

HIDS Host-based Intrusion Detection System

HTTP Hypertext Transfer Protocol

IDS Intrusion Detection System

IoT Internet of Things

IP Internet Protocol

IPS Intrusion Prevention System

IT Information Technology

kNN k-Nearest Neighbors

LDAP Lightweight Directory Access Protocol

ML Machine Learning

MSSQL Microsoft Structured Query Language

NetBIOS Network Basic Input/Output System

NIDS Network Intrusion Detection System

NTP Network Time Protocol

SIEM Security Information and Event Management

SMO Sequential Minimal Optimization

SMOTE Synthetic Minority Oversampling TEchnique

SNMP Simple Network Management Protocol

SSDP Simple Service Discovery Protocol

SVM Support Vector Machine

SYN Synchronize

TCP Transmission Control Protocol

TFTP Trivial File Transfer Protocol

UDP User Datagram Protocol

WEKA Waikato Environment for Knowledge Analysis

ACKNOWLEDGEMENTS

First and foremost, I am very thankful to the Almighty Allah for letting me pursue and fulfill my dreams. I would like to thank my parents from the bottom of my heart for their continuous support throughout my educational and professional career. They have always supported and encouraged me to do my best in all matters of life. They have made immense sacrifices to provide a better life for me and my siblings. I will be forever in their debt. I would also like to take this opportunity to thank my supervisor, Dr. T. Aaron Gulliver for his consistent guidance and support during my educational career. Dr. Gulliver is a very kind, passionate, and approachable person. Without his guidance and patience, this degree would not have been possible. Thank you Dr. Gulliver! My heartfelt thanks to committee member Dr. Mihai Sima and all the University of Victoria staff who contributed in various ways towards the successful completion of this work. My wife Maheen Afridi for her patience and support throughout my studies. My mentor Aman Bukhari for his mentoring, support, and encouragement. Mr. Bukhari provided me with great advice that allowed me to progress in the right direction. Shahbaz Ali, Malek Elgadi, Ahmed Elgarewi, Muhammad Naveed Jokhio, and Salahuddin Jokhio for their time and support. Last but not the least, my friend Wasif Hasan Baig for ramping me up on LaTeX at the very last moment.

Yasar Shahid Hussain

DEDICATION

To my mother and father for their sacrifices, prayers, support, and encouragement.
Thank you for believing in me.

Chapter 1

Introduction

The information age has had a huge impact on traditional industry similar to the industrial revolution. This has allowed the Information Technology (IT) sector to flourish into a multi-billion dollar economy globally. The world has started to see data as the new oil and oil is no longer considered to be the most valuable resource [5]. Enterprises now profit by performing analytics on data such as customer purchasing and voting habits. Governments and private enterprises have invested heavily in IT infrastructure and services which has raised security and privacy issues. Many financial institutions have shifted to online repositories for their business needs. However, this opens new avenues for criminal activities. Cybercriminals or attackers are individuals who use technology to commit malicious acts on digital systems or networks with the intention of stealing sensitive company information or personal data for profit [6]. These malicious acts are called cyberattacks and can vary depending on the intent. Some common examples of these attacks are phishing and Distributed Denial-of-Service (DDoS). The latter remains the number one cyber threat to the availability of business networks, applications, and services [1], and is the main focus of this report.

DDoS is a malicious attempt to disrupt normal traffic to a target server, service or network by overwhelming the target with bogus traffic. The target becomes unavailable to legitimate users because their computing resources are exhausted [2]. DDoS attacks mainly target the application layer which is the human to computer interaction layer where applications access network services. This layer is responsible for producing web pages on servers and delivering responses to Hypertext Transfer Protocol (HTTP) requests. The HTTP protocol allows communications between clients and servers and is a request and response protocol. HTTP requests are not resource intensive on the client slide, but can be for the target server due to database queries or loading files to produce web pages [7]. DDoS attacks are usually carried out by gaining control of infected computers which are commonly referred to as bots. An attacker

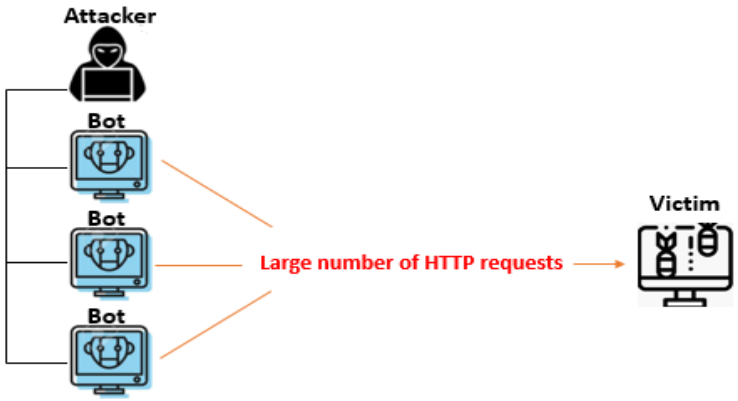

Figure 1.1: Distributed Denial-of-Service (DDoS) attack on a victim [7].

directs the bots by sending instructions remotely via the internet. Figure 1.1 depicts a common DDoS attack scenario [7] where an attacker has control over multiple bots and sends instructions via the internet. The bots send a large number of HTTP requests to the Internet Protocol (IP) address of the victim or target which exhausts their computing resources, resulting in a denial of service to normal traffic. Application layer attacks are difficult to defend against as it can be hard to determine which traffic is malicious because most bots are legitimate internet devices infected with malware [7].

DDoS attacks are a serious threat to the democratic process, information technology and economic sectors [8]. DDoS attacks continue to rise in complexity, volume and frequency, threatening the network security of all enterprises, regardless of their size [1]. DDoS attacks can be categorized into two groups: reflection-based and exploitation-based [9]. Reflection-based DDoS attacks are those in which the identity of the attacker is hidden by utilizing legitimate third-party components such as internet devices to send attack traffic (e.g. HTTP requests) to the victim. These requests are sent to the reflector servers (bots) by the attacker with the source IP address set to the target IP address. These requests are then sent to the victim. Typically, these attacks are carried out using an application protocol, namely Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or a combination of them. TCP based attacks can employ Microsoft Structured Query Language (MSSQL) or Simple Service Discovery Protocol (SSDP) whereas UDP based attacks utilize CharGen, Network Time Protocol (NTP) or Trivial File Transfer Protocol (TFTP) [9]. Certain attacks use a combination of these protocols and include Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), Network Basic Input/Output System (NetBIOS), Simple Network Management Protocol (SNMP), or PORTMAP [9]. Exploitation-based DDoS attacks are also carried

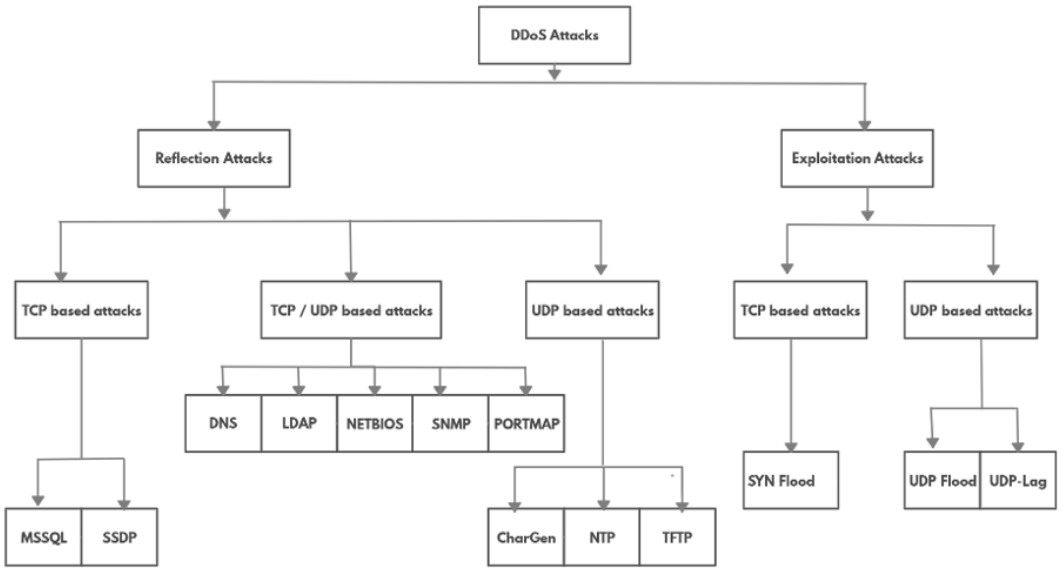

Figure 1.2: Distributed Denial-of-Service (DDoS) attack taxonomy [9].

out using TCP and UDP. TCP based attacks include Synchronize (SYN) flood and UDP based attacks include UDP flood and UDP-Lag [9]. Figure 1.2 provides a DDoS attack taxonomy [9]. Table 1.1 provides a brief description of reflection-based and exploitation-based DDoS attacks.

Class	**Description**
SYN Flood	SYN flood is a denial-of-service attack in which an attacker sends a succession of SYN requests to a target system in an attempt to consume server resources so the system is unresponsive to legitimate traffic [10].
WebDDoS	WebDDoS is an attack to take down the target website or slow it by flooding the network, server or application with bogus traffic [7].
TFTP	A TFTP attack exploits the buffer overflow vulnerability in a Trivial File Transfer Protocol (TFTP) server [11].
DNS	A DNS attack exploits vulnerabilities in the DNS [12].
PORTMAP	PORTMAP is an attack on TCP or UDP port 111 which is a service used to direct clients to the proper port number so they can communicate with the requested Remote Procedure Call (RPC) service [13].
MSSQL	Microsoft Structured Query Language (MSSQL) injection is an attack that makes it possible to execute malicious SQL statements [14].

Class	Description
LDAP	LDAP injection is an attack used to exploit web based applications that construct LDAP statements based on user inputs [15].
NETBIOS	A security exploit in Network Basic Input/Output System (NetBIOS) allows an attacker to see information in computer memory over a network [16].
NTP	NTP is an amplification attack in which the attacker exploits publically-accessible NTP servers to overwhelm the target with UDP traffic [17].
SSDP	An SSDP attack exploits Universal Plug and Play (UPnP) networking protocols in order to send a large amount of traffic to a victim to overwhelm their computing resources [18].
SNMP	A Simple Network Management Protocol (SNMP) attack generates a large amount of traffic which is directed at victims from multiple networks [19].
UDP	User Datagram Protocol (UDP) flooding is an attack in which a large number of UDP packets are sent to a victim with the aim of overwhelming their ability to process and respond. The firewall protecting the target server is exhausted as a result [20].
UDP-Lag	UDP-Lag is an attack that disrupts the connection between the client and server [9].
CharGEN	Character Generator Protocol (CharGEN) flooding is an attack that is carried out by sending small packets carrying a spoofed IP of the victim to internet enabled devices running CharGEN to exhaust computing resources [21].

Table 1.1: Reflection-based and exploitation-based DDoS attacks.

According to CISCO [3], the number of DDoS attacks will almost double to 14.5 million by 2022 as compared to 2017 (see Figure 1.3). They are a serious threat to service providers because DDoS attack size and traffic are increasing at a rapid rate, and the largest attack recorded was 1.7 terabits per second (Tb/s) [1]. In 2018, the cost of downtime associated with outages caused by DDoS attacks was $221,836.80 per attack [1]. Attacks targeting firewalls and Intrusion Prevention System (IPS) devices almost doubled from 16% in 2017 to 31% in 2018 [1]. In this period, there has also been a significant increase in attacks against third-party data centers and cloud services, from 11% to 34%. DDoS remains the number one threat to the

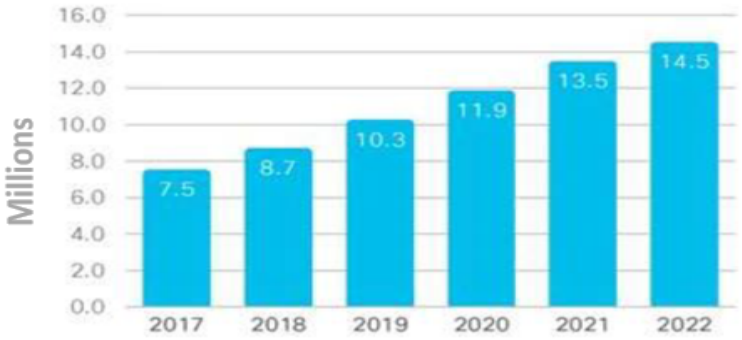

Figure 1.3: Global DDoS attacks forecast, 2017-2022 [3].

availability of business networks, applications, and services. Symptoms of DDoS attack are similar to non-malicious availability issues such as technical problems with the network or system administrators performing maintenance [22]. This poses a challenge to accurately identify (and defend) against these attacks. To identify a DDoS attack, an indication may be slow network performance (accessing files), or unavailability of a particular website [22]. In 2017, the top motivations behind these attacks were criminals demonstrating attack capabilities, gaming, and extortion [1]. Attackers are continuously increasing their computing resources to perform DDoS attacks [23].

1.1 Motivation

Cybercriminals have used DDoS attacks to shut down target servers and infiltrate enterprise networks which can have devastating consequences. Many organizations are ill-equipped to manage modern cyberattacks due to increases in DDoS attack size and complexity. Cybercriminals are aware of the latest technology such as smart devices and the Internet of Things (IoT) and these devices are more susceptible to large-scale DDoS attacks due to their resource limitations (e.g. limited memory and processing capabilities) [2] [24] [7]. In 2016, organizations such as CNN, Netflix, Twitter, Pinterest, and Reddit were offline for nine hours due to an attack on their internet service providers. This resulted in financial losses, loss of productivity, brand damage, credit and insurance rating downgrades, compromised customer and supplier relationships, and IT budget overruns [2].

For as little as $25, cybercriminals can launch a DDoS attack that disables access to a website or server [8]. It is imperative to devise Intrusion Detection Systems (IDSs) to detect DDoS attacks to protect democratic processes, information technology, and economic sectors. Cybersecurity costs can be reduced significantly if technologies such as automation, artificial

intelligence, and machine learning are used by security teams [25]. In this project, Machine Learning (ML) algorithms are used to predict DDoS attacks.

1.2 Related Work

In recent years, significant research has been carried out to mitigate and counter DDoS attacks. Various techniques have been employed such as ML and statistical methods to detect and defend against this threat [9]. However, statistical methods are unable to accurately determine normal network packet distributions whereas ML methods require a feature set for prediction [9]. Many researchers have tried to devise a dataset for this purpose but there are issues such as incomplete traffic, anonymized data, and outdated attack scenarios which limit testing and validating the proposed detection and defense models [9] [26]. A dataset is a collection of instances and attributes (input and output to a model) which is used to train and test an ML model [27]. A dataset is required for all machine learning algorithms. Each instance consists of attributes which can be nominal, numeric, or a string [28]. Machine learning is part of the Artificial Intelligence (AI) paradigm in which systems can automatically learn and improve from experience without being explicitly programmed. The primary goal is to allow the system to learn without human intervention and self-adapt [25]. Machine learning will be further discussed in Chapter 3.

To overcome limitations such as an incomplete dataset, Lashkari et al. [9] devised a comprehensive DDoS dataset called CICDDoS2019 which consists of 80 network traffic features with benign and denial of service flows created using CICFlowMeter. The features were analyzed to obtain the best feature set to detect reflection-based and exploitation-based DDoS attacks. Using this feature set, four common machine learning algorithms were used to predict DDoS attacks, namely ID3, Random Forest, Naïve Bayes, and Logistic Regression. A weighted average of the three evaluation metrics precision, recall, and F-measure was used. Precision is the proportion of correct positive classifications (true positives) from all cases that are predicted as positive. Recall is the proportion of correct positive classifications (true positives) from all cases that are actually positive. F-measure is the weighted harmonic mean of precision and recall and is used as an overall measure of accuracy. In [9], ID3 and Random Forest had the highest accuracy followed by Naïve Bayes and Logistic Regression. In [9], ID3 was chosen as the best algorithm because it had good accuracy and the shortest execution time.

In this project, ML algorithms available in the WEKA tool are evaluated for predicting DDoS attacks. WEKA is a tool used for data analysis and predictive modeling and consists of visualization tools and algorithms [28], [29]. The metrics precision, recall, and F-measure are

determined for the Bayesian Network (BayesNet), Bootstrap Aggregating (Bagging), *k*-Nearest Neighbors (kNN), Sequential Minimal Optimization (SMO), and Simple Logistic algorithms. The CICDDoS2019 dataset from the Canadian Institute for Cybersecurity [4] was modified to have 24 features (see Table 2.1) with 13 class labels (see Table 2.2). The performance of these classifiers is evaluated to identify the best ML algorithm to predict DDoS attacks. The results are discussed in Chapter 4.

1.3 Report Outline

This report is structured as follows.

Chapter 1 provided a brief overview of DDoS attacks including how these attacks are conducted and DDoS attack statistics from the literature. The types of DDoS attacks, evaluation metrics, dataset, and machine learning were introduced. The motivation of the project and related work were discussed.

Chapter 2 presents two types of Intrusion Detection Systems (IDSs) along with two detection mechanisms. The relevant feature and class labels to predict DDoS attacks are also discussed.

Chapter 3 introduces the machine learning tasks along with the parameters used. A brief description of the dataset used in this project is also given. The machine learning tool WEKA is introduced and the most effective classifiers to predict attacks are identified.

Chapter 4 presents the approach employed for intrusion detection along with details of the test environment. First, the hardware and software configuration and evaluation metrics are explained. Then the performance of the Bayesian Network (BayesNet), Bootstrap Aggregating (Bagging), *k*-Nearest Neighbors (kNN), Sequential Minimal Optimization (SMO), and Simple Logistic classifiers is given and discussed.

Chapter 5 concludes this report and summarizes the results. It also provides directions for future work.

Chapter 2

The Intrusion Detection System

An Intrusion Detection System (IDS) is a device or software application that monitors a network for malicious activity or policy violations. It detects vulnerability exploits against a target application or device so appropriate actions can be taken [30]. Most IDSs can be categorized into two groups, Network Intrusion Detection Systems (NIDSs) and Host-based Intrusion Detection Systems (HIDSs). An NIDS is a system which monitors and analyzes incoming network traffic for malicious activity (e.g. SYN flooding) which is then collected centrally using a Security Information and Event Management (SIEM) system. An SIEM system is a software solution that aggregates and analyzes activity from different sources such as network devices, servers, and domain controllers across the entire IT infrastructure. It collects security data and applies analytics to this data to discover trends, detect threats, and enable organizations to investigate any alerts [31]. An HIDS monitors and analyzes the internal components of a computing system (e.g. operating system files) for malicious activity such as the installation of malware [30].

An IDS is based on one of two detection mechanisms, signatures or anomalies [30]. Signature-based detection is based on detecting predefined attacks. Files are mapped to attacks and once matched, an attack type is returned [32]. In the case of a virus, there may be a unique code pattern in a file. If it is discovered again, the file can be flagged as being infected [33]. The benefit of this approach is that the false-positive rate is low [32]. However, modern malware is becoming more sophisticated (polymorphism) so the pattern can change each time the virus is spread from one system to another making it more difficult to detect. Signature-based detection is a reactive approach because it is necessary to have knowledge about a threat in order to develop a signature to detect it [32]. Thus, signatures can only identify threats that are already known. Anomaly-based detection is based on heuristics or rules rather than signatures and attempts to detect behavior that is beyond normal system operation. To identify attacks, a baseline needs

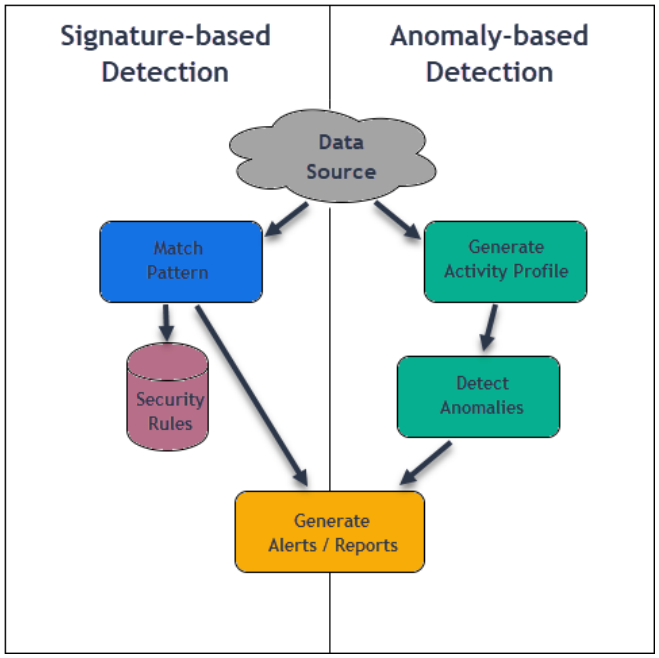

Figure 2.1: Signature-based and anomaly-based detection mechanisms [34].

to be established for the system to recognize normal system activity. This approach can handle unseen or new scenarios, but unfortunately it has a high false-positive rate [32]. Figure 2.1 [34] gives a simplified view of the detection mechanisms. In this project, an anomaly-based NID system is employed. This system considers both reflection-based and exploitation-based attacks based on a feature set and then predicts these attacks using ML algorithms.

2.1 Main Components of the System

The proposed network intrusion detection system employs 24 features (see Table 2.1) to predict 12 different types of DDoS attacks along with benign traffic (see Table 2.2). Once NIDSs predict the attacks, further analysis can be performed by directing the network traffic to centralized data cleansing stations known as scrubbing centers. This attack mitigation system blocks the malicious traffic and allows benign traffic to pass through [35]. The performance and accuracy of the five machine learning algorithms are evaluated based on 5-fold cross-validation which means the dataset is split into 5 groups, and the model is then trained and tested 5 separate times on each individual group to reduce the bias [36]. The evaluation metrics precision, recall, F-measure, True Positive Rate (TPR), False Positive Rate (FPR), and execution time are used to assess the performance of a classifier. Chapter 3 provide details of the machine learning algorithms.

2.2 Features Used in the System

This section presents the 24 best features used in [9] to predict DDoS attacks. The RandomForestRegressor was used to calculate the importance of each feature (out of 80 features) in the dataset. Table 2.1 lists the features which are employed here along with a brief description [37].

Feature	**Description**
Fwd Packet Length Max	Maximum packet size in the forward (outgoing) direction
Fwd Packet Length Min	Minimum packet size in the forward direction
Min Packet Length	Minimum length of a packet
Max Packet Length	Maximum length of a packet
Average Packet Size	Average size of a packet
FWD Packets/s	Number of forward packets per second
Fwd Header Length	Header length of a forwarded packet
Fwd Header Length 1	Number of bytes in a header in the forward direction
Min_Seg_Size_Forward	Minimum segment size in the forward direction
Total Length of Fwd Packet	Packet size in the forward direction
Fwd Packet Length Std	Standard deviation of a packet in the forward direction
Flow IAT Min	Minimum time between two packets in the flow
Subflow Fwd Bytes	Average number of bytes in a sub flow in the forward direction
Destination Port	Address to receive TCP or UDP packets
Protocol	TCP or UDP for data transmission
Packet Length Std	Standard deviation of the packet length
Flow Duration	Duration of the flow in μs
Fwd IAT Total	Total time between two packets in the forward direction
ACK Flag Count	Number of packets with ACK
Init_Win_Bytes_Forward	Number of bytes in initial window in the forward direction
Flow IAT Mean	Mean time between two packets in the flow
Flow IAT Max	Maximum time between two packets in the flow
Fwd IAT Mean	Mean time between two packets in the forward direction
Fwd IAT Max	Maximum time between two packets in the forward direction

Table 2.1: The feature set used in the Intrusion Detection System (IDS) [37].

2.3 Multi-label Classification Used in the System

This section presents the 13 class labels used in the system for attack detection. The following class labels are employed, SYN, TFTP, WebDDoS, DNS, Benign, MSSQL, LDAP, NETBIOS, NTP, SSDP, SNMP, UDP, and UDP-Lag and are described in Table 1.1. These attacks are predicted based on the 24 features listed in Table 2.1.

Chapter 3

Machine Learning

ML is a method of data analysis that automates analytical model building which is a branch of AI based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention [38]. ML and AI based IDSs have been studied extensively during the last decade. The amount of data produced by computing resources such as smart devices and sensors has increased significantly and this allows for detailed analysis which makes ML a popular paradigm. ML allows complex decision-making and problem-solving tasks to be automated such as attack prediction in IDSs. Currently, ML is utilized in many disciplines such as healthcare and engineering [38]. For this project, the dataset CICDDoS2019 was obtained from Canadian Institute for Cybersecurity, University of New Brunswick, Canada [4]. This comprehensive dataset consists of 50063112 instances with 80 features along with 13 class labels to predict DDoS attacks.

3.1 Types of Machine Learning Tasks

There are three types of machine learning algorithms, supervised, semi-supervised, and unsupervised [38], [39]. Supervised learning algorithms attempt to learn the input-output relationship. In classification problems (pattern recognition), the training dataset consists of examples from different classes or labels. The learning task is to construct classifiers that can classify unseen data. These classification problems occur in areas such as object recognition and disease analysis [39]. Unsupervised learning algorithms infer patterns from a dataset without known or labeled instances [38], [39]. They cannot be applied to classification problems because the values for the output data are not known, making it a challenge to train the algorithm [39]. Semi-supervised learning lies between supervised and unsupervised learning. Semi-supervised algorithms are used to solve problems with a small percentage of labeled data and a large per-

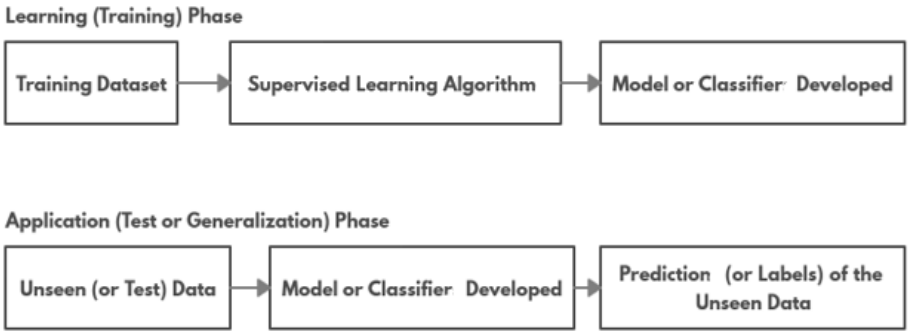

Figure 3.1: The two phases of a supervised learning algorithm [39].

centage of unlabeled data [38], [39]. They are usually utilized in areas such as speech processing, and text and web categorization [39]. This project employs supervised learning algorithms. These algorithms have two phases as shown in Figure 3.1 [39]. The first phase is the learning (training) phase where the algorithm constructs a mathematical model of a classifier based on the training data. The second phase is the application (test or generalization) phase which is used to predict the output for data which was unseen in the learning phase [39].

3.2 Data Splitting

The two most popular approaches to splitting a dataset for training and testing purposes are cross-validation and holdout. Cross-validation is a resampling procedure used to evaluate machine learning models on limited data [39], [40]. It is used to evaluate a machine learning model on unseen data. It uses limited data from the dataset in order to estimate how the model will perform in general when used to make predictions on data not used during training [40]. In this project, k-fold cross-validation is used with $k = 5$ where k is the number of groups that the dataset is split into. Thus, the model is trained on multiple train/test splits. The results are less biased when this approach is used as compared to holdout, since it takes all available data into account [41]. The holdout method is inferior to cross-validation because it uses just one train-test split and the results rely heavily on how the data is split, e.g. 80/20 or 70/30 [40], [41]. Holdout is the simplest and most common method for large datasets as the model only needs to be trained once [40], [41]. One of the drawbacks of this approach is that when the dataset is not balanced (e.g. class imbalance), the training set can be very different from the testing set. This can lead to overfitting and poor performance [40].

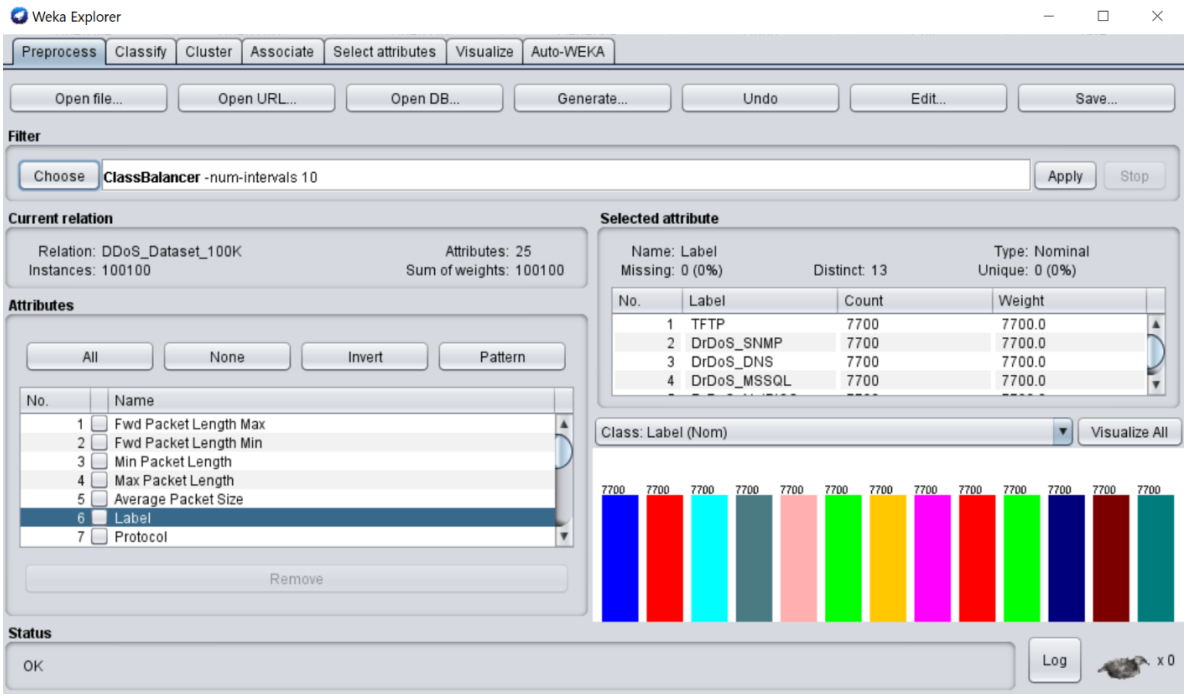

Figure 3.2: The WEKA Explorer Preprocess pane.

3.3 The WEKA Workbench for Machine Learning

WEKA is an open source machine learning tool which is built on the Java platform. It can be accessed through a Graphical User Interface (GUI), standard terminal applications, or a Java Application Programming Interface (API) [29]. It was developed at the University of Waikato, New Zealand and has been widely used for teaching, research, and industrial applications [29]. It consists of supervised, semi-supervised, and unsupervised ML algorithms such as Random Forest, Linear Regression, kNN, and Support Vector Machine (SVM). These algorithms can be modified by tuning their parameters (called hyperparameters) which makes this tool very powerful [28], [29]. Tuning the parameters can increase performance of the algorithms but this is often a trial and error process which is highly dependent on the ML problem [28].

3.3.1 Basic Concepts of WEKA

A dataset is required for all machine learning algorithms. The dataset is input using WEKA Explorer. In the data preprocessing stage, the dataset is selected from the Preprocess pane. Different formats such as .CSV and .ARFF are supported. Figure 3.2 shows a screenshot of the WEKA Explorer Preprocess pane. It provides information on the dataset such as the number of instances and attributes.

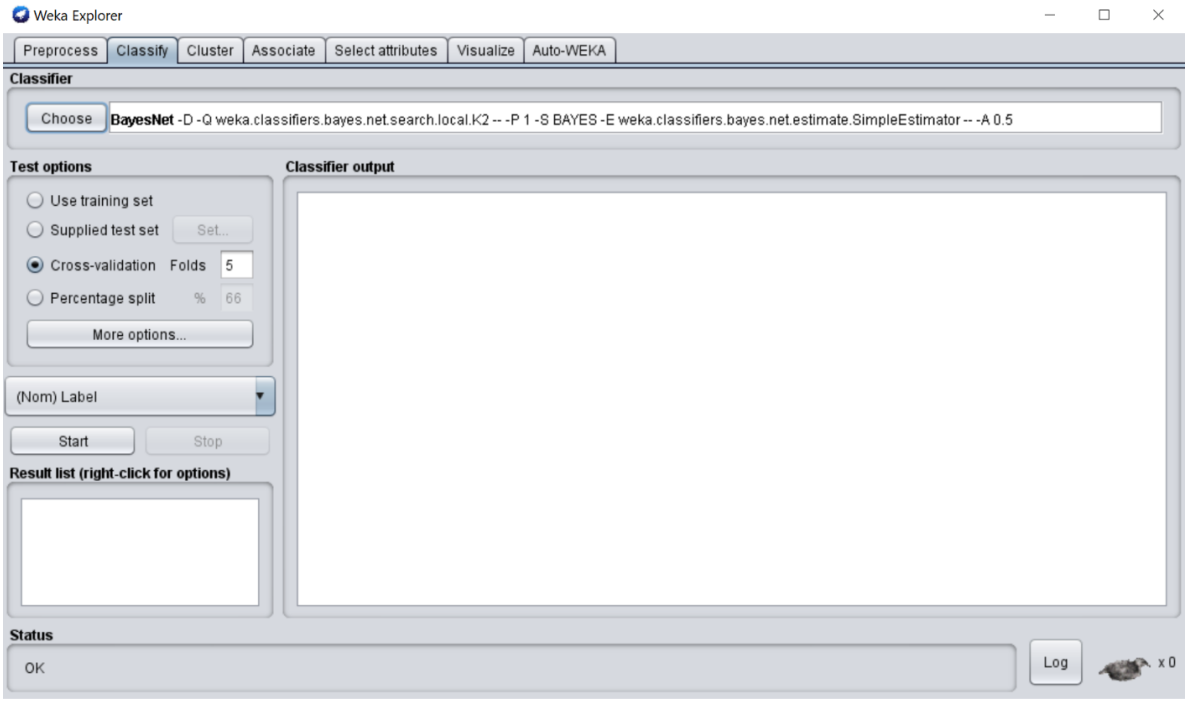

Figure 3.3: The WEKA Explorer Classify pane.

The next step is to select the ML algorithm(s). This is done using the Classify pane. There are various test options supported by the tool, e.g. train/test split and cross-validation. Before building and evaluating a model, the class labels need to be specified. In this project, these labels are the 12 DDoS attacks and legitimate network traffic. Figure 3.3 shows a screenshot of the Classify pane. Once the model is built and evaluated, the results are shown in the Classifier output space. These results are evaluation metrics such as precision, recall, and F-measure along with selected parameters. The results can be exported in various formats such as .TXT. Figure 3.4 shows the results in the Classifier output space.

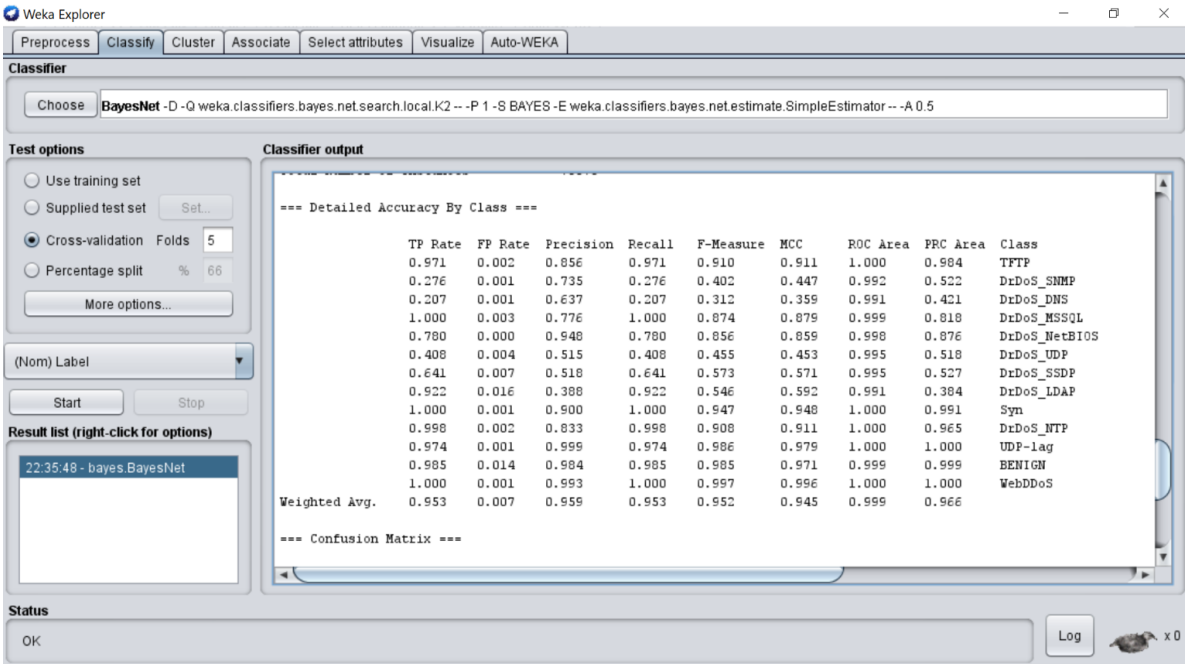

Figure 3.4: The WEKA Explorer Classify pane with classifier output.

3.4 Overview of the Machine Learning Classifiers

Many Intrusion Detection Systems (IDSs) rely on machine learning algorithms to produce accurate results, so it is critical to choose appropriate ones. The following classifiers are used in this project with their default settings, namely Bayesian Networks (BayesNet), Bootstrap Aggregating (Bagging), and Simple Logistic. The k-Nearest Neighbors (kNN) classifier was used with $k = 13$ parameters since the dataset contains 13 class labels. Sequential Minimal Optimization (SMO) for training the SVM was used with LibSVM because it is a popular choice for solving classification problems [39], [42]. Supervised learning classifiers were chosen and evaluated using the dataset. The most popular classifiers (listed below) were selected based on a literature review [39], [43]–[45]. The ML algorithms used in this project are briefly explained below.

3.4.1 Bayesian Network (BayesNet)

The Bayesian Network (BayesNet) ML algorithm is based on the joint probability distribution of a set of random variables with a possible mutual causal relationship [45]. The network consists of nodes representing the random variables, and the edges between pairs of nodes represent the causal relationship of these nodes [45]. The aim of this method is to model the posterior conditional probability distribution of the variables. Bayesian networks are suitable for representing probabilistic relationships and predicting the likelihood of possible causes and contributing factors [45].

3.4.2 Bootstrap Aggregating (Bagging)

The Bootstrap Aggregating ML algorithm is an ensemble method. An ensemble method is a technique that combines predictions from multiple ML algorithms together to make more accurate predictions than an individual model. It sequentially combines weak learners to reduce prediction errors. The algorithm employs a majority vote from a number of bootstrap samples [46]. It is designed to improve the stability and accuracy of the classifier which reduces variances and avoids overfitting. It is often applied to decision tree methods [46].

3.4.3 k-Nearest Neighbors (kNN)

The kNN machine learning algorithm is based on nearest neighbors which is regarded as instance-based learning or lazy learning since it does not learn much from training data [47]. This algorithm is considered to be one of the most effective for classification and regression

problems and it is simple and easy to implement [48]. It is based on feature similarity and employs a majority vote of neighbors. The class with the most votes is taken as the prediction [43], [47]. kNN has some drawbacks such as large storage requirements, high computational complexity in the testing phase, and low tolerance to noise [47].

3.4.4 Sequential Minimal Optimization (SMO)

The SMO algorithm is based on an SVM and uses the LIBSVM library. This algorithm finds the optimal separating hyperplane between two classes by maximizing the closest points between classes [42], [49]. SMO has been used to solve quadratic programming problems that arise during SVM training [42]. SMO breaks these problems into smaller ones which can be solved analytically [42]. It is a powerful algorithm for classification problems but it is computationally expensive to train. It is also sensitive to noisy or erroneous data, so overfitting can be an issue [49].

3.4.5 Simple Logistic

The Simple Logistic machine learning algorithm is based on linear logistic regression. It is widely used due to its simplicity for binary classification problems [49]. This algorithm belongs to the class of generalized linear models which use the logit function. Simple Logistic performs well when the relationship between the data is approximately linear, but performs poorly if a complex nonlinear relationship exists. It also does not handle missing data in the dataset very well [49]. Missing data is defined as data values that are not given for features in the dataset.

Chapter 4

Performance Evaluation

This chapter presents the detection performance of the five supervised machine learning classifiers, namely Bayesian Network (BayesNet), Bootstrap Aggregating (Bagging), *k*-Nearest Neighbors (kNN), Sequential Minimal Optimization (SMO), and Simple Logistic. Six sets of data were created from the CICDDoS2019 dataset based on the number of instances and these were used for training and evaluating the model [4]. All the tests were performed using a laptop with the hardware and software configuration details shown in Table 4.1.

Manufacturer	Dell Inc.
Model	Precision M4600
System Type	64-bit Operating System, x64-based processor
Operating System	Windows 10 Professional
Processor Type	Intel® Core™ vPro™ i7-2860QM
Processor Speed	2.50 GHz
Installed Memory (RAM)	32 GB
Number of Cores	4
Number of Threads	8
Machine Learning Tool	WEKA version 3.8.3

Table 4.1: The hardware and software configuration.

4.1 CICDDoS2019 Dataset

The DDoS attacks in the CICDDoS2019 dataset and the number of instances are given in Table 4.1 [4]. The full dataset contains 50063112 instances including 50006249 DDoS attacks

and 56863 instances of benign (legitimate) network traffic. The 24 features listed in Table 2.1 are used to detect these attacks. The classifiers were trained and evaluated using 5-fold cross validation so the dataset is split into 5 groups. The model is then trained and evaluated 5 separate times.

Attribute (Class Label)	Number of Instances
Benign (legitimate traffic)	56863
DDoS_DNS	5071011
DDoS_LDAP	2179930
DDoS_MSSQL	4522492
DDoS_NetBIOS	4093279
DDoS_NTP	1202642
DDoS_SNMP	5159870
DDoS_SSDP	2610611
DDoS_SYN	1582289
DDoS_TFTP	20082580
DDoS_UDP	3134645
DDoS_UDP-Lag	366461
DDoS_WebDDoS	439

Table 4.2: Number of instances of attributes in the CICDDoS2019 dataset [4].

Due to the size of the CICDDoS2019 dataset and the attribute distribution, it was necessary to modify the original dataset. Table 4.2 shows that the classes are imbalanced, meaning the number of instances differ between classes. This can be a problem when training and evaluating a machine learning model. It can lead to overfitting, for example, DDoS_WebDDoS attack has only 439 instances whereas DDoS_TFTP attack has 20082580 [50].

To overcome the overfitting problem, two techniques are utilized in this project, undersampling and oversampling of the dataset. Undersampling is a simple technique which undersamples large classes randomly. Oversampling is achieved by increasing the minority (smallest) classes using the Synthetic Minority Oversampling TEchnique (SMOTE) [50]. SMOTE is a statistical technique for increasing the number of instances in a dataset such that all class labels have the same number of instances. It generates new instances from existing minority cases [50]. Six sets were created from the CICDDoS2019 dataset and their sizes are shown in Table 4.3. Undersampling was used for the DDoS_1 and DDoS_6 datasets, while both undersam-

pling and oversampling were employed for the DDoS_2, DDoS_3, DDoS_4, and DDoS_5 datasets. The DDoS_1 to DDoS_5 datasets are balanced meaning they have equal number of instances for all 13 class labels while the DDoS_6 dataset is imbalanced as the class label DDoS_WebDDoS had only 30 instances.

Dataset	Total Number of Instances	Benign (normal) Traffic	DDoS Attacks
DDoS_1	5707	439	5268
DDoS_2	26000	2000	24000
DDoS_3	50050	3850	46200
DDoS_4	100100	7700	92400
DDoS_5	739219	56863	682356
DDoS_6	92430	7700	84730

Table 4.3: Six datasets obtained using undersampling and oversampling.

4.2 Evaluation Metrics

An intrusion detection system should have high detection accuracy in predicting DDoS attacks. If the system fails to detect an attack, there can be serious implications for an organization [2]. The evaluation metrics are defined below.

True Positive (tp)–The number of DDoS attacks identified as attacks.

True Negative (tn)–The number of legitimate network traffic instances (benign) identified as legitimate.

False Positive (fp)–The number of legitimate network traffic instances (benign) misidentified as attacks.

False Negative (fn)–The number of DDoS attacks misidentified as legitimate.

Precision is the proportion of true positives from all instances that are predicted as positive and is given by

$$p = \frac{tp}{tp + fp}$$

Recall is the proportion of true positives from all instances that are actually positive and is given by

$$r = \frac{tp}{tp + fn}$$

F-measure is the weighted harmonic mean of precision and recall and is given by

$$f - \text{measure} = \frac{2 \times p \times r}{(p + r)}$$

True Positive Rate is the number of DDoS attacks detected as attacks divided by the number of DDoS attacks in the dataset and is given by

$$TPR = \frac{\sum tp}{\sum \text{DDoS attacks in dataset}}$$

False Positive Rate is the number of benign instances incorrectly classified as DDoS attacks divided by the total number of benign instances in a dataset and is given by

$$FPR = \frac{\sum fp}{\sum \text{Benign traffic in dataset}}$$

4.3 Accuracy of the Classifiers

The DDoS attack detection accuracy was obtained for the Bayesian Network (BayesNet), Bootstrap Aggregating (Bagging), k-Nearest Neighbors (kNN), Sequential Minimal Optimization (SMO), and Simple Logistic classifiers. The results are shown for each of the six datasets, DDoS_1 (Table 4.4), DDoS_2 (Table 4.5), DDoS_3 (Table 4.6), DDoS_4, DDoS_5 (Table 4.7), and DDoS_6 (Table 4.8). Tables 4.9 to 4.14 show the results for each classifier across the datasets. The average performance of the classifiers is then given in Table 4.15.

4.3.1 DDoS_1 Dataset

Table 4.4 shows the accuracy of the five classifiers for the DDoS_1 dataset. The precision, recall, and F-measure are highest for Bagging at 0.961, 0.958, 0.956, respectively, followed by BayesNet at 0.950, 0.944, 0.943, kNN at 0.947, 0.943, 0.940, Simple Logistic at 0.933, 0.915, 0.917, and SMO at 0.899, 0.905, 0.890. The highest TPR (95.79%) was observed for Bagging and the lowest (90.50%) for SMO. Further, Simple Logistic had the longest execution time (8100 s) while Bagging had the lowest (380 s). Bagging provides the best tradeoff between performance and execution time.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
BayesNet	0.950	0.944	0.943	94.37%	5.63%	460
Bagging	0.961	0.958	0.956	95.79%	4.21%	380
kNN	0.947	0.943	0.940	94.26%	5.74%	5840
SMO	0.899	0.905	0.890	90.50%	9.50%	980
Simple Logistic	0.933	0.915	0.917	91.51%	8.49%	8100

Table 4.4: Accuracy results for the DDoS_1 dataset.

4.3.2 DDoS_2 Dataset

The results for the DDoS_2 dataset are shown in Table 4.5. The best classifier for this dataset is kNN with precision, recall, and F-measure at 0.972, 0.967, 0.966 followed by Bagging at 0.970, 0.966, 0.965, BayesNet at 0.965, 0.963, 0.961, SMO at 0.934, 0.936, 0.930, and Simple Logistic at 0.902, 0.898, 0.893. The highest TPR (96.67%) was observed for kNN and the lowest (89.80%) for Simple Logistic. In terms of FPR, kNN had the lowest (3.33%) and Simple Logistic the highest (10.20%). Further, kNN had an execution time of 7074 s which is slightly better than Simple Logistic which took 8931 s. Bagging was the fastest at 616 s followed by BayesNet which took 660 s. Since there is minimal performance difference between kNN and Bagging, the latter provides the best tradeoff between performance and execution time.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
BayesNet	0.965	0.963	0.961	96.32%	3.68%	660
Bagging	0.970	0.966	0.965	96.57%	3.43%	616
kNN	0.972	0.967	0.966	96.67%	3.33%	7074
SMO	0.934	0.936	0.930	93.59%	6.41%	1864
Simple Logistic	0.902	0.898	0.893	89.80%	10.20%	8931

Table 4.5: Accuracy results for the DDoS_2 dataset.

4.3.3 DDoS_3 Dataset

Table 4.6 shows the performance evaluation for the classifiers with the DDoS_3 dataset. In this case, Bagging is the best classifier with precision, recall, and F-measure at 0.968, 0.960, and 0.958, respectively. It is followed by kNN at 0.962, 0.953, 0.951, BayesNet at 0.960, 0.952, 0.949, SMO at 0.943, 0.916, 0.913, and Simple Logistic at 0.939, 0.938, 0.934. The highest TPR (95.95%) was observed for Bagging and the lowest (91.55%) for SMO. In terms of execution time, Bagging is the fastest at 927 s followed by BayesNet at 1183 s. There is minimal performance difference between the classifiers except for the execution time, so Bagging provides the best tradeoff between performance and execution time.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
BayesNet	0.960	0.952	0.949	95.18%	4.82%	1183
Bagging	0.968	0.960	0.958	95.95%	4.05%	927
kNN	0.962	0.953	0.951	95.30%	4.70%	13266
SMO	0.943	0.916	0.913	91.55%	8.45%	2718
Simple Logistic	0.939	0.938	0.934	93.78%	6.21%	29428

Table 4.6: Accuracy results for the DDoS_3 dataset.

4.3.4 DDoS_4 Dataset

Table 4.7 shows the accuracy of the five classifiers for the DDoS_4 dataset. In this case, precision, recall, and F-measure for the kNN classifier are the highest at 0.969, 0.962, 0.961, respectively, followed by Bagging at 0.966, 0.959, 0.958, BayesNet at 0.959, 0.953, 0.952, SMO at 0.947, 0.927, 0.926, and Simple Logistic at 0.916, 0.909, 0.908. The highest TPR (96.19%) was observed for kNN and the lowest (90.91%) for Simple Logistic. Simple Logistic had the highest execution time (41760 s) while Bagging had the lowest (1399 s). There is minimal performance difference between the classifiers except for the execution time, so Bagging provides the best tradeoff between performance and execution time.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
BayesNet	0.959	0.953	0.952	95.30%	4.70%	2022
Bagging	0.966	0.959	0.958	95.93%	4.07%	1399

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
kNN	0.969	0.962	0.961	96.19%	3.81%	20400
SMO	0.947	0.927	0.926	92.70%	7.30%	4128
Simple Logistic	0.916	0.909	0.908	90.91%	9.09%	41760

Table 4.7: Accuracy results for the DDoS_4 dataset.

4.3.5 DDoS_5 Dataset

Table 4.8 shows the performance for the DDoS_5 dataset. kNN is the best classifier, followed by Bagging, BayesNet, Simple Logistic, and SMO. The precision, recall, and F-measure for kNN is 0.982, 0.976, 0.975, Bagging is 0.981, 0.976, 0.975, BayesNet is 0.977, 0.971, 0.970, Simple Logistic is 0.963, 0.962, 0.960, and SMO is 0.965, 0.950, and 0.950. The highest TPR (97.62%) was obtained with kNN and the lowest (95.05%) with SMO. In terms of execution time, Simple Logistic had the highest time (91200 s) while Bagging had the lowest (4200 s). Again, Bagging provides the best tradeoff between performance and execution time.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
BayesNet	0.977	0.971	0.970	97.10%	2.90%	4560
Bagging	0.981	0.976	0.975	97.57%	2.43%	4200
kNN	0.982	0.976	0.975	97.62%	2.38%	79400
SMO	0.965	0.950	0.950	95.05%	4.95%	7820
Simple Logistic	0.963	0.962	0.960	96.20%	3.80%	91200

Table 4.8: Accuracy results for the DDoS_5 dataset.

4.3.6 DDoS_6 Dataset

Table 4.9 shows the performance for the DDoS_6 dataset. In this case, Bagging is the best classifier with precision, recall, and F-measure at 0.970, 0.963, and 0.961, respectively. It is followed by kNN at 0.969, 0.962, 0.960, BayesNet at 0.962, 0.931, 0.941, SMO at 0.945, 0.929, 0.924, and Simple Logistic at 0.931, 0.905, 0.913. The highest TPR (96.27%) was obtained with Bagging and the lowest (90.54%) with Simple Logistic. In terms of execution time, Bagging was

the lowest at 1232 s followed by BayesNet at 1951 s. There is minimal performance difference between the classifiers except for the execution time. Thus, Bagging provides the best tradeoff between performance and execution time.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
BayesNet	0.962	0.931	0.941	93.14%	6.86%	1951
Bagging	0.970	0.963	0.961	96.27%	3.73%	1232
kNN	0.969	0.962	0.960	96.19%	3.81%	19781
SMO	0.945	0.929	0.924	92.88%	7.12%	3870
Simple Logistic	0.931	0.905	0.913	90.54%	9.46%	39952

Table 4.9: Accuracy results for the DDoS_6 dataset.

4.3.7 BayesNet Across the Six Datasets

The BayesNet classifier had the best performance with the DDoS_5 dataset as shown in Table 4.10. The worst performance in terms of TPR was obtained for the DDoS_6 dataset. The highest execution time (4560 s) was with the DDoS_5 dataset while the lowest was with the DDoS_1 dataset at 460 s. The DDoS_1 dataset provides the best tradeoff between performance and execution time. This is due to the size of the dataset which provides comparable results and the lowest execution time, as the execution time depends on the size of the dataset.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
DDoS_1	0.950	0.944	0.943	94.37%	5.63%	460
DDoS_2	0.965	0.963	0.961	96.32%	3.68%	660
DDoS_3	0.960	0.952	0.949	95.18%	4.82%	1183
DDoS_4	0.959	0.953	0.952	95.30%	4.70%	2022
DDoS_5	0.977	0.971	0.970	97.10%	2.90%	4560
DDoS_6	0.962	0.931	0.941	93.14%	6.86%	1951

Table 4.10: Accuracy results for BayesNet across the six datasets.

4.3.8 Bagging Across the Six Datasets

The performance of Bagging was the best for the DDoS_5 dataset as shown in Table 4.11. The worst performance in terms of TPR was obtained with the DDoS_1 dataset. The highest execution time (4200 s) was with the DDoS_5 dataset while the lowest was with the DDoS_1 dataset. The DDoS_1 dataset provides the best tradeoff between performance and execution time. This is due to the size of the dataset which provides comparable results and the lowest execution time, as the execution time depends on the size of the dataset.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
DDoS_1	0.961	0.958	0.956	95.79%	4.21%	380
DDoS_2	0.970	0.966	0.965	96.57%	3.43%	616
DDoS_3	0.968	0.960	0.958	95.95%	4.05%	927
DDoS_4	0.966	0.959	0.958	95.93%	4.07%	1399
DDoS_5	0.981	0.976	0.975	97.57%	2.43%	4200
DDoS_6	0.970	0.963	0.961	96.27%	3.73%	1232

Table 4.11: Accuracy results for Bagging across the six datasets.

4.3.9 kNN Across the Six Datasets

The performance of the kNN classifier is shown in Table 4.12. In terms of TPR, the worst performance was obtained with the DDoS_1 dataset but this provided the fastest execution time (5840 s). The best performance was with the DDoS_5 dataset but the execution time (79400 s) was the highest. The DDoS_1 dataset provides the best tradeoff between performance and execution time. This is due to the size of the dataset which provides comparable results and the lowest execution time, as the execution time depends on the size of the dataset.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
DDoS_1	0.947	0.943	0.940	94.26%	5.74%	5840
DDoS_2	0.972	0.967	0.966	96.67%	3.33%	7074
DDoS_3	0.962	0.953	0.951	95.30%	4.70%	13266
DDoS_4	0.969	0.962	0.961	96.19%	3.81%	20400
DDoS_5	0.982	0.976	0.975	97.62%	2.38%	79400

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
DDoS_6	0.969	0.962	0.960	96.19%	3.81%	19781

Table 4.12: Accuracy results for kNN across the six datasets.

4.3.10 SMO Across the Six Datasets

Table 4.13 shows the performance of SMO with the six datasets. In terms of TPR, the worst performance was obtained with the DDoS_1 dataset at 90.50% but this provided the fastest execution time (980 s). The best performance was with the DDoS_5 dataset at 95.05% but the execution time (7820 s) is the highest. The DDoS_1 dataset provides the best tradeoff between performance and execution time. This is due to the size of the dataset which provides comparable results and the lowest execution time, as the execution time depends on the size of the dataset.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
DDoS_1 Dataset	0.899	0.905	0.890	90.50%	9.50%	980
DDoS_2 Dataset	0.934	0.936	0.930	93.59%	6.41%	1864
DDoS_3 Dataset	0.943	0.916	0.913	91.55%	8.45%	2718
DDoS_4 Dataset	0.947	0.927	0.926	92.70%	7.30%	4128
DDoS_5 Dataset	0.965	0.950	0.950	95.05%	4.95%	7820
DDoS_6 Dataset	0.945	0.929	0.924	92.88%	7.12%	3870

Table 4.13: Accuracy results for SMO across the six datasets.

4.3.11 Simple Logistic Across the Six Datasets

Table 4.14 shows the performance of Simple Logistic with the six datasets. In terms of TPR, the worst performance was obtained with the DDoS_2 dataset at 89.80% but this provided one of the lowest execution times (8931 s). The best performance was with the DDoS_5 dataset at 95.05% but this had one of the highest execution times (91200 s). The DDoS_1 dataset provides the best tradeoff between performance and execution time. This is due to the size of the dataset which provides comparable results and the lowest execution time, as the execution

time depends on the size of the dataset.

Dataset	Precision	Recall	f–measure	TPR	FPR	Execution Time (s)
DDoS_1 Dataset	0.933	0.915	0.917	91.51%	8.49%	8100
DDoS_2 Dataset	0.902	0.898	0.893	89.80%	10.20%	8931
DDoS_3 Dataset	0.939	0.938	0.934	93.78%	6.21%	29428
DDoS_4 Dataset	0.916	0.909	0.908	90.91%	9.09%	41760
DDoS_5 Dataset	0.963	0.962	0.960	96.20%	3.80%	91200
DDoS_6 Dataset	0.931	0.905	0.913	90.54%	9.46%	39952

Table 4.14: Accuracy results for Simple Logistic across the six datasets.

4.3.12 Average Performance of the Classifiers

The average performance of the classifiers is shown in Table 4.15. These results were obtained using Tables 4.11 to 4.15. For each table, the average was calculated by taking the sum of each column namely, precision, recall, F-measure, TPR, and FPR and then dividing it by the total number (six) of datasets. These results show that Bagging has the best average performance. The average precision, recall, F-measure, TPR, and FPR for Bagging are 0.969, 0.964, 0.962, 96.35%, and 3.65%, respectively. Simple Logistic is the worst performing classifier with the lowest average results for precision, recall, F-measure, TPR, and FPR at 0.931, 0.921, 0.921, 2.12%, and 7.88%, respectively.

Classifier	Average Precision	Average Recall	Average f–measure	Average TPR	Average FPR
BayesNet	0.962	0.952	0.953	95.24%	4.77%
Bagging	0.969	0.964	0.962	96.35%	3.65%
kNN	0.967	0.961	0.959	96.04%	3.96%
SMO	0.939	0.927	0.922	92.71%	7.29%
Simple Logistic	0.931	0.921	0.921	92.12%	7.88%

Table 4.15: Average performance of the classifiers across the six datasets.

4.3.13 Comparison of Balanced and Imbalanced Datasets

As mentioned earlier, the DDoS_6 dataset is imbalanced as the class label DDoS_WebDDoS has only 30 attack instances. To evaluate the effect of balancing a dataset, this is compared with the DDoS_4 dataset which is slightly larger as the number of instances of the DDoS_WebDDoS attack is the same as for the other 12 class labels. In terms of TPR and FPR, BayesNet and Simple Logistic performed better (except for the execution time) with the DDoS_4 dataset (Table 4.7) whereas Bagging and SMO had better performance with the DDoS_6 dataset (Table 4.9). For example, TPR and FPR for BayesNet are 95.30% and 4.70% for the DDoS_4 dataset compared to 93.14% and 6.86% for the DDoS_6 dataset. kNN had similar results for both datasets at 96.19% and 3.81%, respectively. Bagging provided the best overall results across both datasets which are 95.93% and 4.07% for the DDoS_4 dataset and 96.27% and 3.73% for the DDoS_6 dataset. The DDoS_4 dataset had a higher execution time for the five classifiers which was expected due to the higher number of instances compared to the DDoS_6 dataset. The execution time for Bagging was 1399 s for the DDoS_4 dataset and 1232 s for the DDoS_6 dataset.

4.4 Discussion

The results in Table 4.9 to 4.14 are weighted averages over all 13 class labels. The execution time was approximately linear with the dataset size, e.g. DDoS_1 dataset had 5707 instances and the execution time was 380 s for Bagging, while DDoS_5 dataset had 739219 instances which took 4200 s. The classifiers had the best performance (except for execution time) with the DDoS_5 dataset because it is the largest. In this case, the ML model is better able to learn during the training phase due to the higher number of instances in the dataset, resulting in better prediction.

Chapter 5

Conclusions

In this project, the accuracy of an IDS to detect DDoS attacks was evaluated. The dataset employed was CICDDoS2019 from the Canadian Institute for Cybersecurity which is the most current and comprehensive dataset available [4]. This dataset was split into six different datasets based on the number of instances using undersampling and oversampling. From [9], the 24 best features were selected to predict DDoS attacks using five supervised machine learning classifiers, namely Bayesian Network (BayesNet), Bootstrap Aggregating (Bagging), *k*-Nearest Neighbors (kNN), Sequential Minimal Optimization (SMO), and Simple Logistic. All classifiers were trained and evaluated using 5-fold cross validation. In order to evaluate the accuracy, the precision, recall, F-measure, TPR, and FPR metrics were used. The results obtained show Bagging provides the best performance and also the lowest execution time. The average precision, recall, F-measure, TPR, FPR and execution time for Bagging were 0.969, 0.964, 0.962, 96.35%, 3.65%, and 1459 s, respectively, followed by kNN at 0.967, 0.961, 0.959, 96.04%, 3.96%, and 24294 s, BayesNet at 0.962, 0.952, 0.953, 95.24%, 4.77%, and 1806 s, SMO at 0.939, 0.927, 0.922, 92.71%, 7.29%, and 3564 s, and Simple Logistic at 0.931, 0.921, 0.921, 92.12%, 7.88%, and 36562 s. Simple Logistic had the highest execution time. Overall, it was concluded that Bagging provides the best tradeoff between performance and execution time. Bagging is an ensemble method that allows multiple models (including weak learners) to be trained. These models then solve the problem by combining several decision trees to produce better predictive performance. This leads to better decisions and more accurate results [51].

5.1 Future Work

For future work, other supervised and also unsupervised machine learning algorithms such as Recurrent Neural Networks and Convolutional Neural Networks could be explored using the

CICDDoS2019 dataset. Real-time packets can be collected and tested against the classified training dataset. The holdout technique for splitting the data could be used to compare with the performance of the classifiers using k-fold cross validation. Increasing the k-value in cross validation may have an impact on classifier performance so it could also be explored.

Bibliography

[1] C. Hildebrand, *Cloud in the crosshairs*, Mar. 2019. [Online]. Available: https://www.netscout.com/blog/cloud-crosshairs (visited on 01/20/2020).

[2] Deloitte, *Defending against distributed denial of service (DDoS) attacks*. [Online]. Available: https://www2.deloitte.com/ca/en/pages/risk/articles/DDoSattacks.html (visited on 01/07/2020).

[3] Cisco, *Cisco annual internet report*, Feb. 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (visited on 01/14/2020).

[4] Canadian Institute for Cybersecurity, *DDoS evaluation dataset (CICDDoS2019)*, 2019. [Online]. Available: https://www.unb.ca/cic/datasets/ddos-2019.html (visited on 01/17/2020).

[5] "The world's most valuable resource is no longer oil, but data," *The Economist*, [Online]. Available: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data (visited on 03/08/2020).

[6] G. Grispos, "Criminals: Cybercriminals," in *Encyclopedia of Security and Emergency Management*, L. R. Shapiro and M.-H. Maras, Eds., Springer, 2019, pp. 1–7.

[7] Cloudflare, *What is a distributed denial-of-service (DDoS) attack?* [Online]. Available: https://www.cloudflare.com/en-ca/learning/ddos/what-is-a-ddos-attack/ (visited on 01/09/2020).

[8] Canadian Centre for Cyber Security, "Cyber threats to Canada's democratic process," Tech. Rep., Aug. 2018. [Online]. Available: https://cyber.gc.ca/en/ (visited on 01/21/2020).

[9] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *International Carnahan Conference on Security Technology*, Chennai, India, Oct. 2019, pp. 1–8.

[10] O. C. Ibe, *Fundamentals of Data Communication Networks*. Hoboken, NJ, USA: Wiley, Nov. 2017.

[11] FortiGuard, *TFTP server buffer overflow*. [Online]. Available: https://fortiguard.com/encyclopedia/ips/10268 (visited on 01/09/2020).

[12] *The most popular types of DNS attacks*. [Online]. Available: https://securitytrails.com/blog/most-popular-types-dns-attacks (visited on 01/06/2020).

[13] D. Smith, *Portmapper is preying on misconfigured servers to amplify attacks*, Sep. 2015. [Online]. Available: https://blog.radware.com/security/2015/09/portmapper-preying-on-servers/ (visited on 01/25/2020).

[14] Akamai, *Attackers using new MS SQL reflection techniques*, Feb. 2015. [Online]. Available: https://blogs.akamai.com/2015/02/plxsert-warns-of-ms-sql-reflection-attacks.html (visited on 01/08/2020).

[15] J. M. Alonso, R. Bordon, M. Beltran, and A. Guzman, "LDAP injection techniques," in *IEEE Singapore International Conference on Communication Systems*, Guangzhou, China, Nov. 2008, pp. 980–986.

[16] Microsoft, *MS03-034: Flaw in NetBIOS could lead to information disclosure*, Sep. 2019. [Online]. Available: https://support.microsoft.com/en-us/help/824105/ms03-034-flaw-in-netbios-could-lead-to-information-disclosure (visited on 01/16/2020).

[17] Cloudflare, *NTP amplification DDoS attack*. [Online]. Available: https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/ (visited on 01/09/2020).

[18] M. Majkowski, *Stupidly simple DDoS protocol (SSDP) generates 100 Gbps DDoS*, Jun. 2017. [Online]. Available: https://blog.cloudflare.com/ssdp-100gbps/ (visited on 01/17/2020).

[19] Imperva, *SNMP reflection/amplification*. [Online]. Available: https://www.imperva.com/learn/application-security/snmp-reflection/ (visited on 01/22/2020).

[20] F. Lau, S. Rubin, M. Smith, and L. Trajkovic, "Distributed denial of service attacks," in *IEEE International Conference on Systems, Man and Cybernetics*, Nashville, TN, USA, Oct. 2000, pp. 2275–2280.

[21] JavaPipe, *35 types of DDoS attacks (that hackers will use against you in 2020)*, Jun. 2019. [Online]. Available: https://javapipe.com/blog/ddos-types/ (visited on 01/11/2020).

[22] CISA, *Understanding denial-of-service attacks*, Nov. 2019. [Online]. Available: https://www.us-cert.gov/ncas/tips/ST04-015 (visited on 01/08/2020).

[23] D. Makrushin, *The cost of launching a DDoS attack*, Mar. 2017. [Online]. Available: https://securelist.com/the-cost-of-launching-a-ddos-attack/77784/ (visited on 01/12/2020).

[24] N. R. Fulbright, *Legal implications of DDoS attacks and the Internet of Things (IoT)*, Dec. 2016. [Online]. Available: https://www.dataprotectionreport.com/2016/12/legal-implications-of-ddos-attacks-and-the-internet-of-things-iot/ (visited on 01/18/2020).

[25] A. Parisi, *Hands-on Artificial Intelligence for Cybersecurity*. Birmingham, UK: Packt, Aug. 2019.

[26] T. Subbulakshmi, K. Balakrishnan, S. M. Shalinie, D. Anandkumar, V. Ganapathisubramanian, and K. Kannathal, "Detection of DDoS attacks using enhanced support vector machines with real time generated dataset," in *International Conference on Advanced Computing*, Chennai, India, Dec. 2011, pp. 17–22.

[27] J. Brownlee, *Data, learning and modeling*, Dec. 2013. [Online]. Available: https://machinelearningmastery.com/data-learning-and-modeling/ (visited on 01/10/2020).

[28] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse, "WEKA Manual for Version 3-7-8," 2013.

[29] *Waikato Environment for Knowledge Analysis (WEKA)*, University of Waikato, New Zealand. [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka (visited on 01/02/2020).

[30] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, Jan. 2013.

[31] S. Bhatt, P. K. Manadhata, and L. Zomlot, "The operational role of security information and event management systems," *IEEE Security & Privacy*, vol. 12, no. 5, pp. 35–41, Sep. 2014.

[32] R. A. Jamadar, "Network intrusion detection system using machine learning," *Indian Journal of Science and Technology*, vol. 7, no. 48, pp. 1–6, Dec. 2018.

[33] C. Nachenberg, "Polymorphic virus detection module," patent US5696822A, Dec. 1997. [Online]. Available: https://patents.google.com/patent/US5696822A/en (visited on 03/08/2020).

[34] J. Gómez, C. Gil, N. Padilla, R. Baños, and C. Jiménez, "Design of a snort-based hybrid intrusion detection system," in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, S. Omatu, M. P. Rocha, J. Bravo, F. Fernández, E. Corchado, A. Bustillo, and J. M. Corchado, Eds., vol. 5518, Berlin, Germany: Springer, 2009, pp. 515–522.

[35] Radware, *What is a scrubbing center?* [Online]. Available: https://security.radware.com/ddos-knowledge-center/ddospedia/scrubbing-center/ (visited on 04/08/2020).

[36] J. Brownlee, *A gentle introduction to k-fold cross-validation*, May 2018. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/ (visited on 01/15/2020).

[37] Canadian Institute for Cybersecurity, *Network traffic flow analyzer*. [Online]. Available: http://www.netflowmeter.ca/netflowmeter.html (visited on 01/04/2020).

[38] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA, USA: MIT Press, 2020.

[39] T.-M. Huang, *Kernel Based Algorithms for Mining Huge Data Sets: Supervised, Semi-Supervised, and Unsupervised Learning*. Berlin, Germany: Springer, 2009.

[40] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification," in *IEEE International Conference on Advanced Computing*, Feb. 2016, pp. 78–83.

[41] P. Gupta, *Cross-validation in machine learning*, Jun. 2017. [Online]. Available: https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f (visited on 01/10/2020).

[42] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Microsoft Research, Tech. Rep. MSR-TR-98-14, Apr. 1998.

[43] S. Kanj, F. Abdallah, T. Denœux, and K. Tout, "Editing training data for multi-label classification with the k-nearest neighbor rule," *Pattern Analysis and Applications*, vol. 19, no. 1, pp. 145–161, Feb. 2016.

[44] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, "An evaluation of machine learning-based methods for detection of phishing sites," in *Advances in Neuro-Information Processing*, M. Köppen, N. Kasabov, and G. Coghill, Eds., vol. 5506, Berlin, Germany: Springer, 2009, pp. 539–546.

[45] E. Charniak, "Bayesian networks without tears," *AI Magazine*, vol. 12, no. 4, pp. 50–50, Dec. 1991.

[46] J. Brownlee, *Bagging and random forest ensemble algorithms for machine learning*, Apr. 2016. [Online]. Available: https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/ (visited on 01/11/2020).

[47] Z. Zhang, "Introduction to machine learning: k-nearest neighbors," *Annals of Translational Medicine*, vol. 14, no. 11, Jun. 2016.

[48] O. Harrison, *Machine learning basics with the k-nearest neighbors algorithm*, Jul. 2019. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (visited on 01/10/2020).

[49] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proceedings of the Anti-Phishing Working Groups Annual eCrime Researchers Summit*, Pittsburgh, PA, USA, Oct. 2007, pp. 60–69.

[50] R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data," *BMC Bioinformatics*, vol. 14, no. 1, p. 106, Mar. 2013.

[51] A. Nagpal, *Decision tree ensembles- bagging and boosting*, Oct. 2017. [Online]. Available: https://towardsdatascience.com/decision-tree-ensembles-bagging-and-boosting-266a8ba60fd9 (visited on 01/29/2020).