

# Intrusion of Malware (DDoS) Detection in IoT Devices Using Machine Learning on Cyberspace

by

Istiak Al Amin

17201025

Salsabil Lamiya

17201115

Noshin Anjum Sheikh

17201114

S. M. Tanjimul Haque

17301095

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Science

Department of Computer Science and Engineering  
Brac University  
January, 2022

©2022. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Istiak Al Amin

---

Istiak Al Amin  
17201025

Salsabil Lamiya

---

Salsabil Lamiya  
17201115

Noshin

---

Noshin Anjum Sheikh  
17201114

Tanjim

---

S. M. Tanjimul Haque  
17301095

# Approval

The thesis titled “Intrusion of Malware (DDoS) Detection in IoT Devices Using Machine Learning on Cyberspace” submitted by

1. Istiak Al Amin (17201025)
2. Salsabil Lamiya (17201115)
3. Noshin Anjum Sheikh (17201114)
4. S. M. Tanjimul Haque (17301095)

Of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 20, 2022.

Examining Committee:

Supervisor:  
(Member)

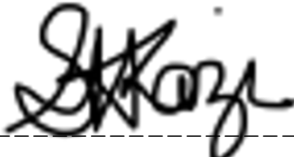
Hossain Arif

-----  
Hossain Arif  
Assistant Professor  
Department of Computer Science and Engineering  
Brac University

Program Coordinator:  
(Member)

-----  
Dr. Md. Golam Rabiul Alam  
Associate Professor  
Department of Computer Science and Engineering  
Brac University

Head of Department:  
(Chair)

  
-----  
Sadia Hamid Kazi  
Chairperson and Associate Professor  
Department of Computer Science and Engineering  
Brac University

# Dedication

We would like to dedicate our research to our beloved family members. Without their unwavering support, encouragement and faith in us, we would not be able to do our task successfully. We would also like to dedicate our humble efforts to all our well-respected teachers.

## Acknowledgement

First and foremost, we are grateful to the Almighty Allah for providing us with the opportunity, direction and guidance to complete this research within time. Secondly, we wish to express our sincerest gratitude to our honorable thesis supervisor Hossain Arif who relentlessly supported, mentored and guided us through a challenging topic. We were able to overcome the obstacles because of their unwavering support and constant feedback. Despite an ongoing pandemic, he managed to spare time for us and offer constructive insights to improve our work and we will forever be grateful for that. Thirdly, we would like to take this chance to thank all of the faculty members for the help and support they have provided in our time in Brac University. Lastly, we would like to express our gratitude towards our beloved parents for their continued prayers, encouragement, and support.

# Table of Contents

Declaration	i
Approval	ii
Dedication	iii
Acknowledgement	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Nomenclature	ix
Abstract	1
<b>1 Introduction</b>	<b>2</b>
1.1 Background . . . . .	2
1.2 Motivation . . . . .	2
1.3 Thesis Orientations . . . . .	3
<b>2 Literature Review</b>	<b>4</b>
<b>3 Research Initials</b>	<b>9</b>
3.1 Research Problem . . . . .	9
3.2 Research Objectives . . . . .	11
<b>4 Methodology</b>	<b>12</b>
4.1 Workflow of the Methodology . . . . .	12
4.2 Description of the Data . . . . .	14
4.2.1 Data Collection . . . . .	14
4.2.2 Data Preprocessing . . . . .	14
4.2.2.1 Data Parsing and Cleaning . . . . .	14
4.2.2.2 Data Normalization . . . . .	15
4.2.2.3 Feature Engineering . . . . .	16
<b>5 Classification</b>	<b>17</b>
5.1 $k$ -Nearest-Neighbors . . . . .	17

5.2	Support Vector Machine . . . . .	18
5.3	Random Forest . . . . .	19
5.4	Naive Bayes Classifier . . . . .	20
5.5	Artificial Neural Network . . . . .	21
<b>6</b>	<b>Implementation of Algorithms</b>	<b>22</b>
6.1	Implementation . . . . .	22
6.1.1	Input Data Preprocessing . . . . .	23
6.2	$k$ -NN Algorithm Implementation . . . . .	23
6.3	SVM Algorithm Implementation . . . . .	25
6.4	Random Forest Algorithm Implementation . . . . .	25
6.5	Naive Bayes Algorithm Implementation . . . . .	26
6.6	Artificial Neural Network Algorithm Implementation . . . . .	26
<b>7</b>	<b>Performance Evaluation of the ML Models</b>	<b>28</b>
7.1	Performance Metrics . . . . .	28
7.2	Confusion Matrices . . . . .	29
<b>8</b>	<b>Experimental Results and Analysis</b>	<b>32</b>
<b>9</b>	<b>Conclusion</b>	<b>35</b>
	<b>Bibliography</b>	<b>37</b>

# List of Figures

4.1	<i>The flow chart of the proposed DDoS detection model. This diagram shows a direct proposition of our work.</i>	13
4.2	<i>Example of Data Normalization.</i>	15
4.3	<i>Algorithm for Data Normalization.</i>	16
5.1	<i>Example of k-NN classifier.</i>	17
5.2	<i>Equation of k-NN classifier.</i>	18
5.3	<i>Example of the piecewise function of SVM classifier.</i>	18
5.4	<i>Example of SVM classifier.</i>	19
5.5	<i>Deriving Kernal for SVM classifier.</i>	19
5.6	<i>Example of Random Forest classifier.</i>	20
5.7	<i>Example of ANN classifier.</i>	21
6.1	<i>A Glimpse of Dataset.</i>	23
6.2	<i>Some Features for Feeding ML Algorithms.</i>	23
6.3	<i>Importing Libraries for Python Coding.</i>	24
6.4	<i>Importing Dataset.</i>	24
7.1	<i>The confusion matrix chart of the k-NN model</i>	29
7.2	<i>The confusion matrix chart of the SVM model</i>	29
7.3	<i>The confusion matrix chart of the Random Forest model</i>	30
7.4	<i>The confusion matrix chart of the Naive Bayes model</i>	30
7.5	<i>The confusion matrix chart of the Neural Network model</i>	31
8.1	<i>Bar Graph on the Accuracy of the ML Model</i>	33



# List of Tables

8.1	PERFORMANCE COMPARISON IN TERMS OF ACCURACY, PRECISION RECALL AND F1 SCORE . . . . .	32
8.2	PERFORMANCE COMPARISON IN TERMS OF AUC AND CON- FUSION MATRIX . . . . .	34

# Nomenclature

The next list describes several symbols and abbreviation that will be later used within the body of the document.

ML Machine Learning  
IoT Internet of Things  
NN Neural Network  
DDoS Distributed Denial of Service  
NB Naive Bayes  
k-NN k-Nearest Neighbor  
RF Random Forest  
LM Levenberg Marquardt  
ANN Artificial Neural Network  
AUC Area Under the Curve  
SVM Support Vector Machine  
ReLU Rectified Linear Activation Unit

# Abstract

Nowadays, the number of interconnected devices (IoT) is increasing dramatically. This expansion poses new security problems for network operators, IoT service providers, and users. Security measures implemented on IoT devices are getting complex due to their heterogeneity and constraints. Attackers have utilized IoT devices to execute massive attacks like DDoS, Zero-Day-Exploitation, Ransomware, etc. The most significant measure to safeguard services from insecure IoT devices is to increase security consciousness in the core network. On the other hand, this thesis suggests a machine learning DDoS detection and diminution technique. The proposed approach was assessed by applying five supervised machine learning classification methods. The evaluation findings reveal that k-NN and Random Forest algorithms outperform ANN, SVM, and Naïve Bayes algorithms. Consequently, the findings of this study can assist academics in further research on malware detection systems for IoT devices.

Keywords: IoT; DDoS; k-Nearest-Neighbour; Random Forest; Naive Bayes; Artificial Neural Network; Support Vector Machine; Cyberspace.

# Chapter 1

## Introduction

### 1.1 Background

Extending the power of the internet beyond computers and smartphones to other environments, genres of devices define the insights of IoT. IoT devices have access to enormous private data, which naturally raises security and privacy concerns. To be more precise, the most concerning attack, for now, is the DDoS attack. Mainly, denial-of-service (DDoS) attackers target network services and websites. The goal is to use enormous traffic of data requests that exceed the network's or server's scope. As a result, the service or website becomes out of reach. With the evolution of threats, ordinary threshold-based methods are useless to correctly classify regular traffic and malicious traffic. These scenarios can be improved and be made more sophisticated by using a more accurate machine learning approach. With the help of a prominent machine learning approach, the accuracy rate of DDoS detection can be improved drastically and the false alarm rate can be suppressed to the minimum. We have proposed a preferable approach to machine learning compared to most shallow learning approaches in our solution. Here, k-NN is used by us as it contains characteristics that are preferred to our objectives. Among those, qualities like simplicity of implementation, efficiency of computation, and high accuracy are preferred. Moreover, we would need to use a weighted tokenized approach to ensure proper pre-training of our solution set. In our proposed solution, a type of support vector machine (SVM) is used to construct a weighted feature set. Finally, the cherished outcome of our proposed work is to get more accuracy from other machine learning approaches.

### 1.2 Motivation

In the early 2000s, the notion of the Internet of Things was initializing, but as we approach 2021, trends reveal that this invention is here to stay. According to reports, by 2021, 35.82 billion IoT devices will be installed worldwide, and by 2025, 75.44 billion [1]. This scenario steamed us most to work on this field firsthand. Though this field is not complete without some threats.

The issue is that DDoS attacks could be extremely vulnerable to the Internet of

Things (IoT) devices. Regular basis software security updates are being lacked. As a result, they are rapidly infected and are being controlled by attackers, which are then used as weapons to attack another internet of things devices. As a result, a machine learning workflow is proposed utilizing binary classification, data gathering and feature extraction in order to monitor the traffic of IoT data. Our solution approach will solve the DDoS attack more efficiently than most shallow learning processes of machine learning.

### 1.3 Thesis Orientations

The following is how the rest of the article is structured. This section provides a brief description of the topics that have been covered in each chapter of this thesis paper. After explaining the purpose of the study and what we hope to accomplish, here the rest of the article is put together in a sequential manner.

- Chapter 2 includes the Literature Review where we summarize related works and information collected from various scholarly articles relevant to this thesis.
- Chapter 3 presents the research objectives in details.
- Chapter 4 includes the methodology suggested for the whole study's workflow.
- Chapter 5 describes five algorithms for our ML model proposed for the study.
- Chapter 6 describes the machine learning algorithms approach to detect DDoS attacks.
- Chapter 7 evaluates the performance of the ML models.
- Chapter 8 outlines the experimental results and analysis.
- Chapter 9 concludes this thesis with subsequent plans.

# Chapter 2

## Literature Review

In quest of information related to the cybersecurity of IoT sector and DDoS attack, we have gone through some fantastic works done by several scholars. Here we would like to mention some of the gist of their work.

In [10], throughout the IoT network Farhan Ullah et al. introduced a comprehensive approach to detecting pirated and malware infected application. In addition with the deep NN, using source code TensorFlow has offered to identify pirated application. In industrial IoT an architecture model is propose for cybersecurity protection and threats measures. In this system, Farhan Ullah et al has developed four databases in cloud storage so that Farhan Ullah et al may handle malware files and software files which are pirated. According to Farhan Ullah et al the initial database provides new data to the pre-processor to process the new data and apprehend valuable features. From the signatures in databases two and four, Farhan Ullah et al pre-processed data in later submitted to the detection process to capture malware and pirated software. By using an image classification problem with four phases, Farhan Ullah et al. also show that the color images are developed from original binary flies thus they can control the malware detection problem. On the other hand, TFIDF and LogTF weighting techniques focus the importance of each token and screen the noisy data regarding plagiarism in the original code. Moreover, color image visualized deep CNN is used in order to detect malicious IoT network infections, which contains five modules. First and foremost, the convolution layer is used by Farhan Ullah et al. in order to reduce noise and improve the signal properties. Secondly, Farhan Ullah et al. uses the pooling layer in order to reduce data overhead while still retaining important information. After that, Farhan Ullah et al. utilized a fully connected layer to turn the two-dimensional array into a one-dimensional array, which is subsequently sent on to the appropriate classifier for further processing. The classifier is used by Farhan Ullah et al. to identify malware families from respective photos in the fourth and final step. Farhan Ullah et al. conclude by showing interest in working for unknown malware families in the future.

For detecting false alarm rate, scalability and accuracy Diro A. Abebe and Chilamkurti N. in [9] proposed a novel distributed learning technique of DDoS attack detection in fog-to-things computing where they claim their experiments have shown that shallow models are less superior than deep models. To be more precise, the fog-to-things network is especially the network of user-end intelligent devices designed

as distributed intelligence; we may also use them as responsive hosts of security services, analyzing big data, etc. Diro A. A. and Chilamkurti N. then claimed that, like other facilities, fog-to-things devices have always faced plenty of vulnerabilities such as probing, DoS, ransomware, smurf, etc attacks due to limited resources, which can be overcome by using time-window-based statistics and previous n-connection based approach. Moreover, they compared saying that, in attack detection, classical machine learning approaches lack several features; thus, DL with its stacked autoencoder (SAE) is introduced, enriched by its automatic hierarchical feature and pre-trained (such as softmax regression) to determine actual attack more efficiently. After reviewing many papers, Diro A. A. and Chilamkurti N. conclude that the main objective of their paper is to employ distributed deep learning through parameters and model exchange where most of the employed DL architecture has successfully used pre-training for feature extraction. They also claim that the SGD (stochastic gradient descent) for fog-to-things computing is not practical, so they proposed the multi-fold technique, which can be defined by the master IDSs responsible for revamping parameters using GD (gradient descent); at the same time, the workers IDSs diagnose harmful events locally. Furthermore, the data set used here is called the NSC-KDD dataset and the examination process was conducted on Keras on Theano DL package with the Apache Spark framework. The process they followed is encoded the categorical features into the discrete features and by using a stacked autoencoder, all the hidden features were extracted by applying softmax algorithm classification. Diro A. A. and Chilamkurti N. clarified test results denoting that they found accuracy: 99.20% and detection rate (DR) is about 99.27%, where shallow learning shows DR of 97.50%, which results from a clear win situation of DL approach. They have concluded with an expectation of investigating its performance on different data sets and other neural networks.

The paper [4] starts with discussion on Distributed Denial of Service (DDoS) attacks' severities. This type of attack can be used to deplete the bandwidth supplies for crucial WA (web applications), causing these services inaccessible to users or even banning internet connections for a significant geographic portion, resulting in significant economic losses. In this paper, Kai Wang et al. aims at significantly improving the security of data sharing, and then the security of the whole network, even the whole CPS by design a Secure Networking architecture, combining blockchain and AI together (named as SecNet). SecNet, a blockchain-based data sharing mechanism was introduced, where any sharing ready data need to be registered into a blockchain, named Data Recording Blockchain (DRB). In addition, only DRB can examine the genuineness and probity of data. Besides, SecNet ensures financial incentives between different entities for sharing data and exchanging security service. In this way, data security and encouraging data sharing throughout the CPS is executed by SecNet.

In [12], Chih-Ta Lin et al. state that many infrastructure production facilities, enterprise management systems (EMS) are widely utilized. In order to, monitor production operations using industrial network control protocols, most existing public industrial installations use a human-machine interface (HMI) such as the SCADA system and the programmable logic controller (PLC). Due to the lack of authentication and encryption mechanisms in this system, it is effortless for hackers to penetrate the network and release attack commands. Previous related work has

always focused on securing the network layers, third layer from OSI model, but it couldn't identify penetration attacks such as spoofing attacks. This paper focuses more on cyber incursions in the OSI model (the data link layer). It proposes a two-phase intelligent intrusion detection method to record the typical behavior patterns, and later it is used to detect abnormal behavioral events in developed ICS test bed. Firstly, in the training phase, all network communication has monitored using a data packet parser, and using machine learning statistical analysis approach, they establish the system's normal behavior patterns. In the Detection phase, collected normal behavioral patterns used to detect the system's anomalous behavior. In conclusion, they suggested focusing more on the machine learning phase to make classification-based learning methods to detect malware intrusion in the future.

This paper [14] presents a detailed methodology for threat identification and prevention during the implementation of medical devices. During medical device deployment, this article focuses on risk assessment, threat detection and management. In the existence of unknown security threats, it is important to maintain solitude and security; IoT Devices can detect and analyze risk significantly, following that they take automatic action plans when the risk is raised. Here, in order to provide limitless threat elimination during device activation, Aakarsh Rao et al. introduced a threat detector with real-time action with a dynamic risk estimation technique Markov Models are used here for threat detection. These models use cumulative distribution functions during run time, collecting both normal and abnormal behavior of medical IoT devices. As a result, they developed a probabilistic approach to risk analysis. To evaluate the current system risk in framework design Aakarsh Rao et al. used Composite Risk Model, which will be revised significantly. Threat reduction method either disables access to the influenced component or changes the current functioning method to limit the risk while maintaining critical operations. To identify the critical operations, the software application is probabilistically studied and described in the device's combined infection model to build the normal execution model, all for per window Risk Assessment and Management Unit for window N. To explain the architecture, the Aakarsh Rao et al. approaches a smartly-linked pacemaker prototype and injects malicious virus into it.

In the beginning of [8], Mohamad Syahir Abdullah et al. figured that with the appearance of ransomware attacks, along with malevolent cyber activities done by individuals and governments, an instructive savvy is needed. In order to process those enormous volumes of data, the writer proposed to classify the data and figure out a model made of a word-level feature set and a sentence-level form to obtain more precision. Furthermore, it also should be noticed that by advancing in the cyber sector, new terms need to be included, and classifiers like Named Recognition Entity, CRF, and LSA need to be forced to eliminate data redundancy. After that, the Mohamad Syahir Abdullah et al. combinedly have worked on and explored different techniques like NER, CRF, etc for making efficient data retrieving mode. With unstructured global data, the term weighting approach using SVM and NB was used. In addition to that have used the LSA technique to solve ambiguity problems, where Mohamad Syahir Abdullah et al. have used Porter Stemmer to pre-process text. After that, NER is used in order to find a keyword that was integrated manually via python programming. At this point, CRF-classifier will be trained to classify news and an approach called f-measure will be used to measure the information processing.



To conclude, as ongoing research, the researchers have created the feature set for cyberattacks and the CRF classifier with the LSA approach is yet to be done.

In this study [11], Rohan Doshi et al. using IoT-specific network behaviors to guide feature selection, Rohan Doshi et al. argued that it can lead to high accuracy DDoS detection in IoT network traffic using some ML algorithms, including NN. Moreover, Rohan Doshi et al. develop a machine learning pipeline designed to monitor network middleboxes (such as routers, firewalls, and network switches) for unusual gridlock and associated machines that could be part of a bot-net. There are four steps in the anomaly detection pipeline. Firstly, the traffic capture process will record all data packets sent from smart home devices, including the source port, source IP address, destination port, destination IP address, timestamp and packet size. To operate as a middlebox, they set up a Raspberry Pi v3. Rohan Doshi et al. then capture regular (non-DoS) traffic. On the other hand, to overcome the challenges of collecting DoS traffic, Rohan Doshi et al. proposed Kali Linux VM as a source. They collected victim and victimizers data from an Apache Web Server executed by Raspberry Pi 2 which was considered the DoS victim. Secondly, Each IoT device's packets are split by their originating IP address. Thirdly, stateless and stateful characteristics are generated for each packet based on domain knowledge about IoT device behavior in the Feature Extraction part. Fourthly, in the BC (binary classification) part, K-NN, decision trees, SVM, RFC and DNN can accurately distinguish between DoS attack traffic and normal traffic. Rohan Doshi et al. conclude by showing the interest in research into ML malware detection in the future to safeguard networks against vulnerable IoT devices.

In paper [13], Hafiz M. Farooq and Naif M. Otaibi introduced numerous models for utilizing Machine Learning analytics to improve Cyber security monitoring alongside detecting many other common cyber threats using an optimal algorithm. This paper compared K-Means, DBSCAN, BIRCH to analyze upload and download traffic crucial for detecting cyberattacks. In windows process execution, they used kernel-based classification approaches such as one class support vector machine (OCSVM) for analyzing anomalous windows registry entries and used logistic regression to do transaction analysis. Moreover, a linear regression algorithm uses to compare abnormal activity with behavioral baseline. Lastly, for message classification, the paper showed to use of a random forest classifier ML algorithm. To conclude, OCSVM (one class SVM ) are easier to train, less expensive, and more appropriate to allowing SOC Analysts to conduct originality detection and discover new signs of understanding.

Merging these ideas, in the IoT sector, DDoS assaults are most vulnerable and individuals become the hostage of attackers in the worst-case scenario [5]. Acknowledged by the severity of DDoS attacks in IoT devices from [9], [11] and [4], it was our clear choice to work on the detection of intrusions of malware in IoT devices.

We have got our initial idea from [10], where Farhan Ullah et al. have proposed an image processing approach to detect piracy on software. The image processing approach was pretrainable and it introduced us to deep learning (DL). According to [9], the deep learning approach is more accurate and the intrusion detection rate (DR) is 99.27% prominent from the ML approach, which is 97.50%. Regardless of prominence, we chose to implement our malware detection system with ML because

deep learning usually needs a significant amount of training data to ensure that the network contains millions of parameters to exclude overfit the data. According to [11], in order to obtain faster results, machine learning methods are preferable. They are easier to train and use less processing power, where deep learning models take time to train. As a result, ML was our clear choice over deep learning.

In order to observe different algorithms of machine learning, we studied [13] and become aware of clusters, classifiers and probability models to implement machine learning. Among all other techniques, the K-means clustering and OCSVM classifier captured our attention the most. The K-means clustering works faster than most other clustering algorithms such as DBSCAN, BIRCH. On the other hand, Hafiz M. Farooq and Naif M. Otaibi, from [13], stated that OCSVM is easier to train and less expensive in parallel to other Decision Trees or Naïve Bays classifiers.

To make our approach the most accurate based on a more user-friendly environment, we decided to use these two methodologies in our paper.

# Chapter 3

## Research Initials

### 3.1 Research Problem

With so many possibilities, IoT technology is expected to advance well beyond anyone's wildest expectations. However, as IoT devices become more prevalent, there will be an increase in IoT application development as well as security concerns and difficulties. By 2019, global internet expenditures are predicted to reach \$745 billion, up 15.4% from \$646 billion in 2018. By the end of the projected period in 2022, the market will have surpassed the \$1 trillion worth of market value [3]. This massive number of devices will produce vast sets of data that consist of different structures and will be impactful in so many ways in human advancement. In order to tackle those vulnerabilities, the term cybersecurity is introduced. Cybersecurity is the process of ensuring protocol in all devices and virtual platforms connected to the internet to prevent unwelcome appearance, access blockage, illegal control in personal space, data stealing etc. DDoS attacks, which adversely affect not only internet infrastructure but also its applications, which are amongst the most destructive forms of network attacks, and they are becoming more common. In rare cases, attackers might employ this form of attack to drain the bandwidth supply for popular and critical online applications, leaving these services inaccessible to consumers or even prohibiting internet access for a large area of a nation, causing severe economic losses. As a consequence of everyone on the internet exchanging security rules, we will see a significant reduction in the victimization that DDoS attackers may exploit. This will result in a more thorough understanding of network security [4].

DDoS attacks are carried out through the use of networks of computers that are connected to the internet. These networks are made up of compromised IoT devices that an attacker may remotely manipulate. Bots are individual devices; however a botnet is a group of bots that work together to attack a target [5]. If an attacker has successfully constructed a botnet, he or she may conduct an attack by sending remote commands to each bot in the network. Whenever a botnet attacks a victim's system or network, every bot sends queries to the user's IP address, possibly overwhelming the system or network and enabling it to become unavailable to normal traffic. Although each bot is a legitimate internet device, it might be difficult to distinguish between infected and regular traffic.

With the aim of obtaining an efficient solution for DDoS attacks, machine learning techniques are proposed. Learning via inference and patterns without explicitly programming using algorithms and statistical approaches is known as machine learning (ML). This last decade has seen a huge advancement in machine learning technology. We proposed a strategy based on five fundamental techniques of ML to cope with DDoS attacks on IoT devices. The techniques, we proposed to apply here, are k-NN, SVM, Artificial Neural Network, Random Forest and Naïve Bayes.

The first technique to be applied on our problem will be k-NN classifier. There are various classification methods available, including k-means, Linear Regression and Decision Tree etc. k-NN algorithm has been proven to be very successful for a wide range of problem areas, including text categorization and classification. It seems to be using the class labels of a test sample and k of its neighbors to determine the class label for the test example. The similarity score amongst each k nearest document and also the test document is determined in order to establish the weights assigned to the categories in a k - nearest document. It has been successfully used to the calculation of the distance between neighbors.

The Random Forest algorithm comes next, and this is a sanctioned learning approach that merges classification trees in the same method that each tree is dependent on the very same random variable of the sample vector. For its great performance and simplicity, Random Forest is often utilized for classification and regression. The random forest algorithm makes easy to understand predictions. It processes huge datasets well. The random forest algorithm outperforms the decision tree method in predicting outcomes. This algorithm calculates the result based on the predictions made by the decision tree. It makes predictions by taking the average of the findings from different trees. When we increasing the number of trees precision also improves.

Furthermore, In terms of classification algorithms, the Nave Bayes Classifier is among the most basic and productive options accessible today. It aids in the creation of quick machine learning methods that can give correct predictions in a short timeframe. The field of machine learning has made considerable strides in current years, and it has shown itself to be not just simple, but also reliable. This is a classification algorithm, which means that it generates predictions estimate the probability of an item existing in a certain situation. The Bayes theorem has been at the heart of this classifier. It has been effectively applied to a variety of tasks, but it performs particularly well when dealing with problems involving natural language processing (NLP), real time prediction etc.

Artificial neural networks (ANNs) are computer programs that employ learning techniques to adapt to the information they receive. Thus, they are excellent for non-linear statistical data modeling applications like as AI systems with deep learning capabilities are important in machine learning as well as the larger subject of AI. There are three or more layers in an artificial neural network. The first layer is made up of input neurons. They transmit information to deeper layers, that in turn transmit information to the peripheral output layer. Inside the outer layers are units that translate the intelligence gathered from one layer to the next, which are concealed from view. It is possible for the ANN to comprehend more intricate things thanks to the input and output layers. It is all of these convolutional nodes that

are together referred to as the neural layer. Practical applications include corporate intelligence, spam email identification, natural language processing in automation, and a variety of other applications.

The other method to apply here will be a type of linear regression model called Support Vector Machine which uses a space to divide data into two groups. This space can be linear, Gaussian, non-linear, sigmoid, polynomial etc depending on the function used in the model. Using more than one space, data can be splitted into more than two classes by SVMs. We are using this methodology to break them down into component classes such as HTTP, FTP, SMTP, etc and evaluate internet gridlock patterns [6]. In part due to the fact that SVM is a supervised machine learning approach, it is commonly used in applications that may imitate attacks, such as those that utilize network traffic produced during testing process as training data.

## 3.2 Research Objectives

The primary objective of this study is to create an intrusion detection system for detecting DDoS attacks by combining classifier machine-learning approaches such as k-NN, Random Forest, Naïve Bayes, ANN along with support vector machine (SVM). Typically, an IoT device receives a variety of data and delivers it to a central system via gateways for additional processing. The proposed methodology could identify malware in data sent by IoT devices. The proposal of our research is summarized in the followings:

1. This research utilizes a number of different machine learning models. We employed five models to identify DDoS attack patterns, and we found that they were all accurate.
2. Certain Machine Learning algorithms' performance has produced excellent findings, which are noticeably better than those obtained from previous research conducted on this dataset.
3. This paper provides an overview of a comparative analysis of the performance of some Machine Learning models that were used in the development of this thesis.
4. Appraising the detection model with different dataset and more combination of deep learning algorithms to gain more accuracy.

# Chapter 4

## Methodology

### 4.1 Workflow of the Methodology

By observing the packets and traffic parameters that change unusually in each phase of the attack, we study the procedures of the DDoS attacks. As a result, we are in need of a method that acquires more specific results with more accuracy. In this present work, mainly we have discussed five classifier algorithms such as k-Nearest-Neighbors (k-NN), Naïve Bayes, Random Forest, Support Vector Machine (SVM) and Artificial Neural Network (ANN). In order to work with these advanced machine learning algorithms, first we need to collect a dataset or arrange one from a simulated DDoS attack. After that, we need to work on the dataset as there exists tons of redundant data which need to be got rid of or merged with some meaningful values. Moreover, it is necessary to choose the premier features from the dataset by which it would be possible to figure out the malicious behavior of that data. After figuring out the features, the dataset needs to be trained by the machine learning approach. The figure 1, depicts the comprehensive view of our detailed work.

1. Dataset Preparation: Our data collection process resulted in the creation of a comma separated value (csv) file. After reading the dataset, it was transformed to a data frame using the Panda library for easy display and analysis.
2. Data Preparation: The data gathered in the preceding step may include duplicate records, unavailable data or noisy data. As a result, it is essential to understand the various characteristics of the data. At this point, the data has been split into a collection of distinct features.
3. Training and learning over the normal dataset: In this step, we would divide the dataset into two parts [60:40] for training and testing our model.
4. Testing and evaluation: Finally, our proposed model needs to be evaluated for accuracy.

As we have used two machine learning algorithms to build two different DDoS detection models, here we would run an accuracy test for more efficient results.

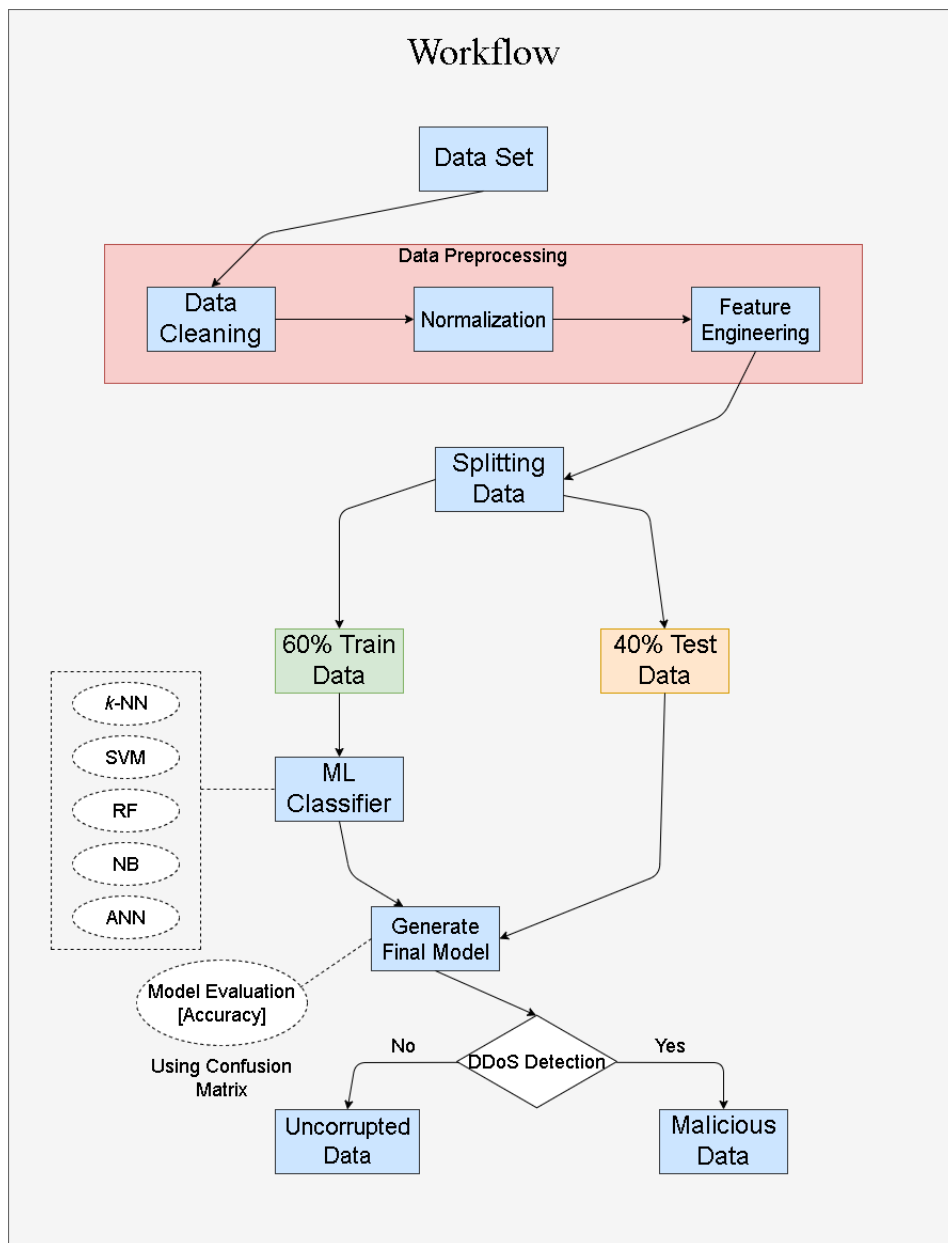


Figure 4.1: The flow chart of the proposed DDoS detection model. This diagram shows a direct proposition of our work.

## 4.2 Description of the Data

### 4.2.1 Data Collection

Primary collection and secondary collection are the two most common methods of gathering data and putting it together into a dataset. Primary data collection is the process of gathering information directly from the field while doing research. Researchers start by analyzing and collecting data from the scratch.

Secondary data collection is the process of gathering information from previously conducted studies, tests, and primary data in order to continue their study. For the purpose of our study, we have collected our data through the use of secondary data sources.

The unprocessed network packets (Pcap files) in the BoT-IoT dataset were produced in the Cyber Range Lab of the Australasian Center for Cyber Security (ACCS) by the use of the T-Shark program. The dataset contains a mixture of normal and anomalous traffic. Ostinato and Node-red were used to create simulated network traffic for testing purposes (for non-IoT and IoT respectively). A variety of source files for the dataset are given, including the original Pcap files, the produced ARGUS files, and ultimately a CSV formatted version. It was decided to divide the files depending on attack category and subcategory in order to make the labeling procedure easier. In this dataset, there are over 1 million entries and 47 columns.

### 4.2.2 Data Preprocessing

Machine Learning requires preprocessing of data before fitting any model since machines are only capable of understanding numerical data. During this step, we start preparing our data to be used for the learning processes and for the extraction of unique patterns. Many components are included in this step, including data cleaning, data normalization, feature selection, feature extraction, and the division of the data into training and testing datasets. By preprocessing, we can extract the clean and original data. There are some steps for preprocessing.

#### 4.2.2.1 Data Parsing and Cleaning

The information gathered in the prior phase may include duplicate records, missing information, or noisy information. As a result, it is essential to understand the various characteristics of the data. At this point, the data has been segmented into a number of distinct features. Each feature includes a large amount of data. Data redundancy needs to be removed so we need to replace any items that have been lost on average. In addition, as mentioned in algorithm1, it is essential to remove and exclude any duplicate information. The data is then saved in a database, and the values of each feature are calculated using the minimum, maximum, mean, and standard deviation values.



### Algorithm 1 - Data Cleaning

#### Function:

Input →(DS) Mixed of normal and abnormal (malicious) Datasets

Output →Return cleaned and transformed Datasets

#### Data Cleaning:

array of data →DS.csv

for i: array of data. Length

Remove Redundant data

End for loop

#### End of Data Cleaning

#### 4.2.2.2 Data Normalization

Data may include a range of values, a varied mean, and a variable variance, all of which contribute to learning difficulties and reduce the efficiency and accuracy of the learning process. As stated in algorithm2, we utilized the min-max scaling method to mitigate the detrimental effect of marginal values. Thus, all data values in the range 0 to 1 are represented. This method is often referred to as feature scaling, and its formula is given in eq1.

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}} \quad (4.2)$$

Figure 4.2: *Example of Data Normalization.*

## Algorithm 2 - Data Normalization

### Function:

Input  $\leftarrow$  (DS) Mixed of normal and abnormal (malicious) Datasets  
Output  $\leftarrow$  Return cleaned and transformed Datasets

### Data Normalization:

Array of Features  $\leftarrow$  feature extraction  
for i: Array of Features. Length  
    for j: array of data[i]. Length  
        min  $\leftarrow$  minimum value  
        max  $\leftarrow$  maximum value  
        new transformed value calculated from equation 1  
    End for loop  
End for loop

### End of Data Normalization

Figure 4.3: *Algorithm for Data Normalization.*

### 4.2.2.3 Feature Engineering

On the other hand, we might look at two groups of characteristics and examine why they are important in distinguishing between regular and malicious IoT data. Individual packet attributes that are not reliant on the flow may be used to generate stateless characteristics. These characteristics are produced without the need to segment the incoming traffic stream according to its IP source. As a result, these characteristics are the most lightweight. Stateful variables are used to record the evolution of network traffic over time. Because we divided network data into brooks per device and divide the per-device streams into time windows, there is an inherent cost in producing these features. As a basic time series, the time windows serve the representation of the devices' changing network activity, which is represented by the time windows. These characteristics need aggregating data over many packets inside a time frame; although the middle-box conducting classification must maintain state, the quantity of state retained may be reduced by utilizing brief time windows.

# Chapter 5

## Classification

In order to evaluate system performance and identify unexpected occurrences that are incompatible with typical network behavior, machine learning techniques are used. Even in network systems with large amounts of data flowing, anomalous behaviors may be recognized by mathematical models built using machine learning techniques, and preventative measures can be implemented in real time within those networks. The characteristics of the machine learning methods that were applied in this research are briefly discussed in this section.

### 5.1 $k$ -Nearest-Neighbors

Classifier algorithm  $k$ -NN has the characteristics that will be used to identify DDoS attacks and categorize the network state into several categories [15]. Following that, we consider assigning the present network state to one of the classes. For document classification, there are many well-known techniques available including NN, rough set and fuzzy logic. We pick the  $k$ -NN technique because it has characteristics that are conducive to our objectives. These characteristics include ease of implementation, efficient calculation, and high accuracy. One of the similarity-based learning algorithm called the  $k$ -NN has shown to perform very well in a different of problem areas, including classification issues. The  $k$ -NN method locates a test element  $dt$ 's  $k$ -nearest-neighbors among the training components that define  $dt$ 's neighborhood. The class for  $dt$  is determined by majority vote among the neighborhood components. As shown in Fig. 5.1, we begin by identifying the  $k$  elements that are closest to the sample to be classified. We select the best appropriate class for the test element by examining the  $k$  nearest items.

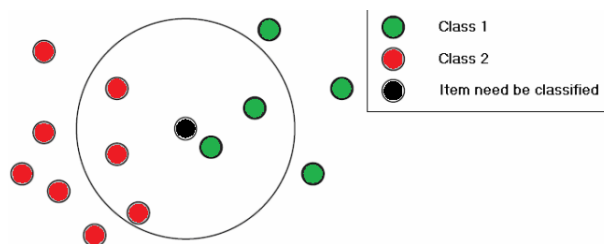


Figure 5.1: *Example of  $k$ -NN classifier.*

The word 'near' can also be defined as degree to which two components are comparable. There are many methods for determining the degree of similarity between two components. However, the technique based on the cosine formula is the most often used method for calculating the degree of similarity. This method is used in this research to calculate the similarity degree. Additionally, we define each element using the vector space model (VSM). As a result, each element is represented as an n-component vector. An example is given as follows. For the two components  $X \equiv \{x_1, x_2, \dots, x_n\}$  and  $Y \equiv \{y_1, y_2, \dots, y_n\}$ ,  $W \equiv \{w_1, w_2, \dots, w_n\}$  denotes the weighted vector and  $w_i$  denotes the weight of the component  $i$  in the general vector. After that, computing the similarity between two elements  $X$  and  $Y$  as follows:

$$\text{Similarity (X, Y) = Cosine (X, Y, W) = } \frac{\sum_{i=1}^n (x_i \times w_i) \times (y_i \times w_i)}{\sqrt{\sum_{i=1}^n (x_i \times w_i)^2} \sqrt{\sum_{i=1}^n (y_i \times w_i)^2}} \quad (5.1)$$

Figure 5.2: Equation of k-NN classifier.

We may determine the  $k$  nearest elements using the cosine method stated before. Following that, we must identify the most appropriate class for these elements. We count the number of iterations of each class type to determine which has the maximum rate. This is the class that may carry the test element.

## 5.2 Support Vector Machine

When only "normal" datasets are provided and no boundaries are known, the SVM method developed by SchÖlkopf et al. [16] is sufficiently well fitted for originality identification scenarios. Therefore, we collected our 'normalized' data and then put it into the SVM classification algorithm.

As an example, SVM maps the input dataset into a hyper-dimensional variable space  $H$  and iteratively searches for the hyper-space with the highest margin that best splits the data subsets from the source.

$$f(x) = \begin{cases} +1 & \text{if } x \in S \\ -1 & \text{if } x \in \bar{S} \end{cases} \quad (5.2)$$

Figure 5.3: Example of the piecewise function of SVM classifier.

In our scenario, let  $x_1, x_2, \dots, x_l$  be training samples belonging to a single class  $X$ , where  $X$  is a compact subset of the  $\mathbb{R}^n$ . Let  $\phi : X \rightarrow H$  be a kernel map that converts the training samples into a different space. Afterwards, in order to distinguish the data set from the origin, it is necessary to solve the quadratic programming problem outlined below:

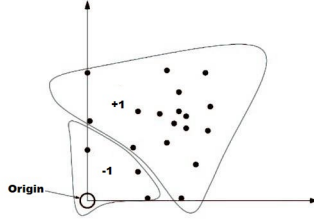


Figure 5.4: *Example of SVM classifier.*

$$\min \frac{1}{2} \|\omega\|^2 + \frac{1}{vl} \sum_i \xi_i - \rho$$

$$\text{Subject to } (\omega, \phi(x_i)) \geq \phi - \xi_i, \xi_i \geq 0. \quad (5.3)$$

If  $\omega$  and  $\rho$  solve this problem, then the decision function

$$f(x) = \text{sign}(\omega, \phi(x)) - \rho \quad (5.4)$$

Figure 5.5: *Deriving Kernal for SVM classifier.*

will be positive for most models  $x_i$  included in the training set.

During the study of command line parameters, we utilized a variety of Kernels, such as Gaussian, Linear, Polynomial, Radial, to analyze the data. When using SVM, choosing the most "suitable" kernel with the most acceptable kernel parameters is a very essential part that is highly dependent on the particular task at hand. When compared to other kernels, such as 2-degree polynomial, radial and additionally available kernels, the linear kernel provided more "stable" decision boundaries and produced a greater number of confirmed anomalies throughout our study.

### 5.3 Random Forest

Technically, it is an ensemble technique (based on the divide-and-conquer strategy) of decision trees created on a randomly divided dataset that is applied to a large number of decision trees. The forest is a set of decision tree classifiers that is used to classify data. Information gain, gain ratio, and Gini index are used to create the separate decision trees for each variable. Each tree is based on a distinct arbitrary model. In a classification problem, the class with the most votes is picked as the final output when each tree casts a single vote. On basis of regression, the end result is the standard of all the tree outputs, which is calculated as the final result. RN is both simpler and more powerful to utilize comparing to the other non-linear classification algorithms.

Suppose the RF classifier is  $R(x)$ ; decision tree  $i$  is denoted as  $t(x)$ ,  $R(x) = t_i(x)$ ,  $i \in [0, n - \text{estimators}]$ , where the number of decision trees in the RF is

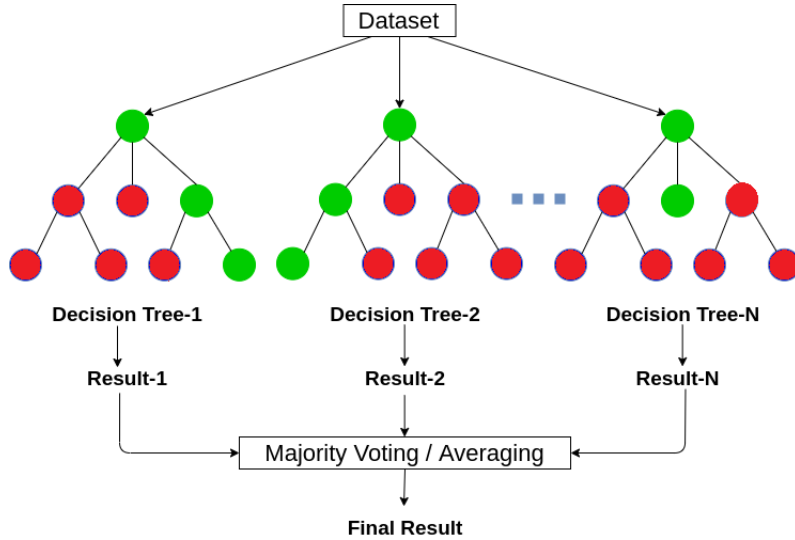


Figure 5.6: Example of Random Forest classifier.

represented by  $n$ -estimators,  $x$  is the input training sample for classification, and  $sign(x) \in S$  is the tag value of  $x$ , in which  $S$  is the set of labeled attributes, the output of the  $t_i(x)$  is a constant value in  $S$ , and the output of the  $R(x)$  is the mode of the estimated value of  $t_i(x)$ ,  $i \in [0, n - estimators]$ . While RF is used for testing,  $x$  is from the new training dataset which is randomly generated by resampling method in the attribute training set; there should be only two kinds of labeled data in DDoS attack-detection, which define attack or normal. To mark the attack sample labels and normal sample labels,  $S \equiv \{ 1, 0 \}$ , and  $sign(x)$  can only take the value 1 or 0, respectively.

## 5.4 Naive Bayes Classifier

The Naive Bayes classifier is the most basic kind of probabilistic classifier, and it is most often used. In the case of probabilistic classifiers, the output  $\Pr(\text{Event1} | \text{Event2})$  represents the probability that the data  $\text{Event2}$  belongs to the category  $\text{Event1}$ . Each data includes terms by which probabilities have been assigned depending on the number of times they have appeared in that specific dataset. The pattern of evaluating a collection of test data that has been properly classified and, therefore, comparing the contents in all categories may be learned via supervised training, and Naive Bayes can learn this pattern by developing a list of features and the probability of their occurrence. As a result, a list of data appearances may be used to categorize new data into the appropriate categories based on the greatest posterior probability distribution.

Bayes theorem is stated mathematically as the following equation:

$$f(\text{Event1} | \text{Event2}) = \frac{P(\text{Event2}) | P(\text{Event1}) * P(\text{Event1})}{P(\text{Event2})} \quad (5.5)$$

Where:  $\text{Event1}$  and  $\text{Event2}$  are events

$P(\text{Event1})$  and  $P(\text{Event2})$  are the probabilities of  $\text{Event1}$  and  $\text{Event2}$  independent of each other

$P(\text{Event1} | \text{Event2})$ , a conditional probability, is the probability of  $\text{Event1}$  given that  $\text{Event2}$  is true

$P(\text{Event2}|\text{Event1})$ , is the probability of Event2 given that Event1 is true

## 5.5 Artificial Neural Network

An Artificial Neural Network (ANN) is a kind of information processing paradigm that is constructed up of a large number of densely linked processing components (neurons) that work together to solve particular problems. ANNs, like humans, learn by observing and imitating. An artificial neural network (ANN) is trained to perform a certain task, such as pattern recognition or data categorization, using a learning process. Because of their exceptional capacity to extract patterns from intricate or inaccurate data, neural networks may be used to extract patterns and discover trends from data that would be impossible to detect using other computational approaches. A trained neural network examines the information provided with maximum accuracy. Afterward, this technique may be used to generate predictions for new circumstances that are of interest to the researcher.

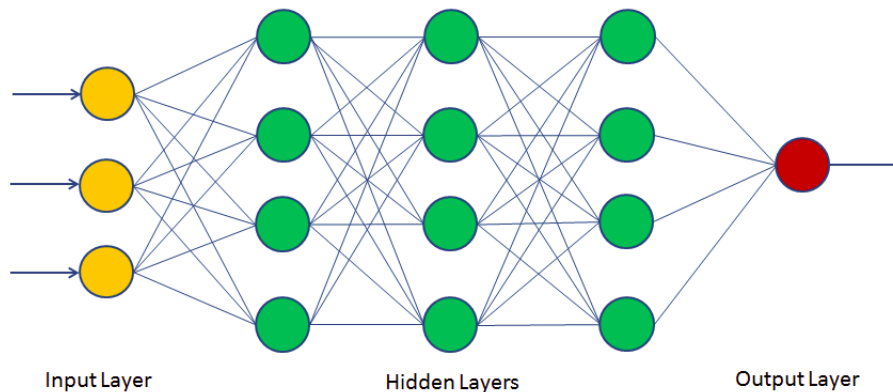


Figure 5.7: *Example of ANN classifier.*

Based on the studies and discoveries made in relation to ANN use models and their advantages, a three-layer ANN feed-forward model has been developed and implemented with a modification of principal component analysis (PCA) for dimensionality reduction in the research. A dimensionality reduction process is provided in order to lower the storage and computation cost of a Levenberg-Marquardt learning algorithm for deriving the Jacobian matrix for large NN models, such as used in the proposed thesis, in order to improve performance. The ANN model used in this implementation is seen in Figure [5.3], in which the decreased measurement input features are supplied to the ANN model before training.

# Chapter 6

## Implementation of Algorithms

### 6.1 Implementation

As we have chosen Python programming language for our work, the Jupyter Notebook was the best choice for implementing the codes for our study. In order to get the work done, we first need to import some libraries of Python such as, Pandas, KNeighborsClassifier, svm, train\_test\_split. For training purposes, we utilize the terms packet, time, source, destination, protocol, and size. On the basis of increasing prediction accuracy, we may or may not select Info fields to include. The same feature set is applied to all of the classifiers in an equal way in order to find the one that performs the best in terms of accuracy.

To get the accuracy from the data that we gathered previously for training purposes we use sci-kit-learn and the pandas which are the Machine Learning libraries. As a result, for training and testing purpose we split the data into two groups, in the proportion of 60:40, and to check for inaccuracies we use a confusion matrix.. Using the confusion matrix, we can determine whether or not a result is accurate by looking at the first diagonal, and whether or not a result is incorrect by looking at the other diagonal, from which we can determine whether or not a result is accurate. Prior to this stage, we must get the data for training in the form of a CSV file, which contains all of the data in a scaled and normalized format. This will need the completion of the data mining phase.

The information gathered by Wireshark must be stored as a PCAPNG file and then converted to a CSV file for further processing. It is not possible to utilize this data for training since the scales of the data are all different. As a result, we must first clean the necessary data, which may be accomplished in such a manner that labels can be provided and text content can be converted to appropriate numbers. There may be some gaps in the data in certain regions. We therefore make use of the notion of binning to enter fake data into the proper places. As a result, the data would be in a variety of scales, which would need to be combined into a single scale before being fed into the activation function of the ML algorithm. We utilized the Pandas library from the Python programming language to do all of this. Following classification, we must plot the accuracy distribution on a graph, which may be accomplished with the help of mat-plot-lib.



## 6.1.1 Input Data Preprocessing

In our research, we have chosen to get data from a secondary source, which is composed by famous experts that have gathered, tested, and then created a suitable simulated dataset for further investigation. Since the data we had gathered was in a PCAPNG format, it was not clean enough. A CSV file was created next, which we exported. As a consequence, we were able to access the dataset with ease and modify it as required. An example of uncleaned data is given.

No.	Time	Source	Destination	Protocol	Length	Info
4936	12.07511	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2839122 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]
4937	12.07511	117.219.2	192.168.1	TCP	298	443 > 50450 [PSH, ACK] Seq=2840082 Ack=4449 Win=421 Len=244 [TCP segment of a reassembled PDU]
4938	12.07511	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2840326 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]
4939	12.07511	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2841286 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]
4940	12.07511	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2842246 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]
4941	12.07511	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2843206 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]
4942	12.0752	192.168.1	117.219.2	TCP	54	50450 > 443 [ACK] Seq=4449 Ack=2844166 Win=8317 Len=0
4943	12.07685	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2844166 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]
4944	12.07686	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2845126 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]
4945	12.07693	192.168.1	117.219.2	TCP	54	50450 > 443 [ACK] Seq=4449 Ack=2846086 Win=8317 Len=0
4946	12.07887	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2846086 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]
4947	12.07888	117.219.2	192.168.1	TCP	1014	443 > 50450 [ACK] Seq=2847046 Ack=4449 Win=421 Len=960 [TCP segment of a reassembled PDU]

Figure 6.1: A Glimpse of Dataset.

After obtaining the dataset, we processed it using the Python programming language in order to utilize it for further machine learning classifications. After cleaning the dataset, we identified many significant features from the raw data.

Time	Source	Destination	Protocol	Length
12.07511	117.219.2	192.168.1	TCP	1014
12.07511	117.219.2	192.168.1	TCP	298
12.07511	117.219.2	192.168.1	TCP	1014
12.07511	117.219.2	192.168.1	TCP	1014
12.07511	117.219.2	192.168.1	TCP	1014
12.07511	117.219.2	192.168.1	TCP	1014
12.0752	192.168.1	117.219.2	TCP	54
12.07685	117.219.2	192.168.1	TCP	1014
12.07686	117.219.2	192.168.1	TCP	1014
12.07693	192.168.1	117.219.2	TCP	54
12.07887	117.219.2	192.168.1	TCP	1014
12.07888	117.219.2	192.168.1	TCP	1014

Figure 6.2: Some Features for Feeding ML Algorithms.

From here, we will apply machine learning classifiers in order to get result and build a training model.

## 6.2 $k$ -NN Algorithm Implementation

$k$ -NN algorithm is one of few most famous and used algorithms for classification and regression in machine learning. In our study, first we have imported some additional libraries.

After that, we have imported our dataset for preprocessing. During this phase, we prepare large-scale data in order to improve learning processes and the extraction of distinguishing features. Many components are included in this step, such as data

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Figure 6.3: *Importing Libraries for Python Coding.*

cleaning, data normalization, feature engineering and the splitting of the data into training and testing datasets.

```
df_train = pd.read_csv('E:/thesis/DDOS/UNSW_2018_IoT_Botnet_Full5pc_1.csv')
```

Figure 6.4: *Importing Dataset.*

As a preventive measure against over-fitting, we will split our dataset into training and testing divisions. This will provide us with a better understanding in the testing phase that how our algorithm operated all over. As it would be in a production application, our algorithm is tested on unseen data. We divided the dataset as 60 percent of the data being train data and 40 percent being test data. It is usually a good idea to scale the features, in the time of generating any real predictions so that regardless of their relevance, they may all be examined in the same approach.

In a large number of algorithms for machine learning, if the range of values in raw data is too massive then objective functions will not operate correctly. This is because in some machine learning algorithms a range of possible values in raw data is too wide. For example, the Euclidean distance is used by the vast majority of classifiers to compute the distance between two endpoints between them. If one of the features has a large variation, the distance between the two points will be determined by that specific feature. Consequently, the scale of all characteristics should be normalized to say that each feature contributes an equal amount to the final distance.

Normalized to say that each feature contributes an equal amount to the final distance. Furthermore, when features are normalized the gradient descent algorithm is utilized by many other machine learning algorithms which converge more efficiently. Particularly in the time of utilizing the Scikit-Learn library, it seems very simple to train the k-NN algorithm and using it to generate predictions.

Sklearn library is used to import the K-Neighbors Classifier class which is very necessary. The initial step in this process is the Neighbors library. In the second line of code, initialization of this class is done with a single statement, `n_neighbors`. This is basically the value for the `k`. `k` cannot be defined perfectly, and the number that is selected as the best fit is the product of considerable testing and evaluation.

The most often used metrics while evaluating an algorithm is the confusion matrix, accuracy, recall, and f1 score. These metrics may be calculated with the help of the confusion matrix of the sklearn.metrics library. The findings indicate that our k-NN algorithm was capable of recognizing all of the data in the test set with high accuracy.

## 6.3 SVM Algorithm Implementation

To read data from a CSV file `read_csv` function of the `panda's` library is the smoothest and the most elementary method. The preprocessing procedure includes Splitting the data into attributes and labels and dividing the data into training and testing sets.

At first, all of the columns of the "UNSW 2018 IoT Botnet Full5pc 1" dataframe are placed in the `x` variable, which corresponds to the label column. After that, the dataframe is saved in the `y` variable, which corresponds to the label column. The `drop()` function removes all of the unnecessary columns from the table. After that, in the `y` variable only the class column is being stored. The `y` variable has labels to those attributes to which the `x` variable contains.

The last stage in the preprocessing procedure is to separate the data into training and test sets, after all of the data has been separated into attributes and labels. Fortunately, the `train_test_split` method allows us to perfectly divide data into training and test sets which are contained in the model selection library of the Scikit-Learn library.

Now our task is to train the collected SVM on the training data. Scikit-Learn library provides built-in classes for a variety of SVM methods where SVC class is included. we will use the support vector classifier class that is denoted by SVC in Scikit-Learn's SVM library, as we'll be doing a classification job. The type of kernel is the only parameter of this class. Linearly separable data is the only data that basic SVMs can classify that is the reason this is very necessary. We simply set this parameter to "linear" in the first place, in the case of a simple SVM.

For each SVC class, the `fit` method is invoked. By using the training data, the `fit` method trains the algorithm that is given as a parameter to it. To create predictions, the `predict` method of the SVC class is utilized. To represent the accuracy of the data and the outcomes of our study, we further utilized metrics such as the confusion matrix, F1 measure, recall, and precision.

It can be seen from the observations that the SVM method was only marginally outperformed by the k-NN approach.

## 6.4 Random Forest Algorithm Implementation

The Random Forest offers a strong indicator of feature selection because of its random nature. An additional variable is included with the model by Scikit-learn, and this variable indicates the relative importance or contribution of each feature to the prediction. When the training phase is complete, it automatically calculates the relevance score for each feature. Then it reduces the importance to the point where the sum of all values corresponds to 1. In order to begin, we first divided the columns into dependent and independent variables (or features and labels). After that, we divided the variables into two groups: a testing set and a training set.

Utilizing the divided data, we trained the model on the training set and made predictions on the test set, after splitting the data. Following training, it is crucial to compare the accuracy of the model with the actual and projected values. Some features were discarded, such as "id," "saddr," and "sport" features, among others,

because they are of low value. Fifteen features were chosen from the remaining features. A model was created based on specified training set features, predictions were made for selected test set features, and the predicted values were compared with the actual value[17].

Increasing accuracy was achieved by deleting the least important features. That's because decreasing the amount of inaccurate data and noise results in higher accuracy. A smaller number of characteristics also minimizes the amount of time required for training.

## 6.5 Naive Bayes Algorithm Implementation

The Gaussian Naive Bayes classifier is one of the simplest Naive Bayes classifiers to comprehend. According to this classifier, the data from each label is selected from a basic Gaussian distribution, which is assumed to be accurate. Loading our data file into the program was the first thing we did. The data is in CSV format, which means there is no header line and no quotation marks. The open function may be used to open the file, and the reader function to read the data lines from the dataset, in the CSV module. After that, we split the data into two groups: the testing dataset and the training dataset.

The class value divides the mean and standard deviation for each attributes that are included in the summary of the collected training data. These are needed to determine the probability of particular attribute values belonging to each class value in the time of making predictions.

Following the splitting procedure, it is time to create predictions based on the summaries of our training data that have been prepared. it is important to determine the probability that a given data instance belongs to each class, and then to select the highest probabilistic class to make predictions [18]. Finally, our team develop our main function, which executes all of the methods we have explained one after one to regulate the accuracy of our model.

## 6.6 Artificial Neural Network Algorithm Implementation

In order to solve a specific problem the neurons those work together as a combined system are known as Artificial Neural Network(ANN). Neurons communicate with one another and are a collection of connected units or nodes known as Artificial Neuron. An artificial Neural Network consists of artificial neurons. Each link has the capability of transmitting a signal to neighboring neurons. Afterward, these neurons process the signal and transmit it to other neurons that are connected to them. An input signal is a real number. Some non-linear functions' sum of the inputs determines the output of each neuron. Interconnections between two nodes are known as Edges. As the learning process continues, weight changes in which Neurons and edges are often assigned. The weight has an effect that either increases or decreases the strength of the signal at a connection. Neurons may be programmed with a threshold such that if the collection of signals crosses the threshold at that moment a signal is only transmitted. Neurons are generally assembled together into

layers. A variety of changes may be applied to different layers to their respective inputs. There are three types of layers used here. The first one is the input layer where the signals move from then the second one is the output layer where the signals comes from the input layer and another one is hidden layers which are the layers in between input and output layer.

Python is used for writing Deep learning API Keras and machine learning platform TensorFlow is used for running it. In order to facilitate rapid experimentation, it was designed with the goal of allowing researchers to go from idea to result as swiftly as possible. In Keras, layers serve as the fundamental building blocks of neural networks. Layers are made up of two parts in TensorFlow, : method which means a Tensor in Tensor-out computation function and TensorFlow variables as weights store some state. A Layer instance can be called in the same way that a function can be called. Layers, in contrast to functions, keep track of their state, which is updated as the layer gets data during training and is kept in the layer. Weights variable. In the Keras layers API, there are several different built-in layer activation functions. For our computation, we have selected the ReLU function, which is the most often used of the available functions.

The rectified linear unit activation function (ReLU function) is applied using this function. This function returns the typical ReLU activation:  $\max(x, 0)$ , the element-wise maximum of 0, and the input tensor  $x$  where the default values are used[20].

# Chapter 7

## Performance Evaluation of the ML Models

To assess and define a comparative analysis of the results, the F1 score, precision, accuracy, recall, and Confusion Matrix for each of these models have been generated with the goal of evaluating and establishing a comparative analysis of the results for the pre-trained ML models. In this chapter, we will provide the equations for the performance measures that will be applied in this paper. Following that, we illustrate the Confusion Matrices of each of the machine learning models that we have developed so far.

### 7.1 Performance Metrics

The equations of the performance metrics used are specified as follows:

1. Accuracy Formula[19]:

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{P+N} \\ &= \frac{(TP + TN)}{TP+TN+FP+FN} \quad (7.1) \end{aligned}$$

2. Precision Formula[19]:

$$\begin{aligned} \text{PPV} &= \frac{TP \times TPR}{TP + FP} \\ &= 1 - \text{FDR} \quad (7.2) \end{aligned}$$

3. Recall Formula[19]:

$$\begin{aligned} \text{PPV} &= \frac{TP \times TPR}{TP + FN} \\ &= 1 - \text{FDR} \quad (7.3) \end{aligned}$$

4. F1-Score Formula[19]:

$$\begin{aligned} \text{F1} &= 2 \times \frac{PPV \times TPR}{P+N} \\ &= \frac{2 \times TP}{(2 \times TP) + FP + FN} \quad (7.4) \end{aligned}$$

Here, the respective abbreviations are, TP = True Positive, TN = True Negative, P = Positive Case, N = Negative Case, FP = False Positive, FN = False Negative, PPV = Positive Predictive Value, TPR = True Positive Rate, FDR = False Discovery Rate

## 7.2 Confusion Matrices

A classification model's performance can be evaluated and represented using a confusion matrix. Therefore, utilizing the test data, we have created confusion matrices for each of the Machine Learning models used in this thesis. Here, all the figure below depict the Confusion Matrices for all of the ML models studied in this paper.

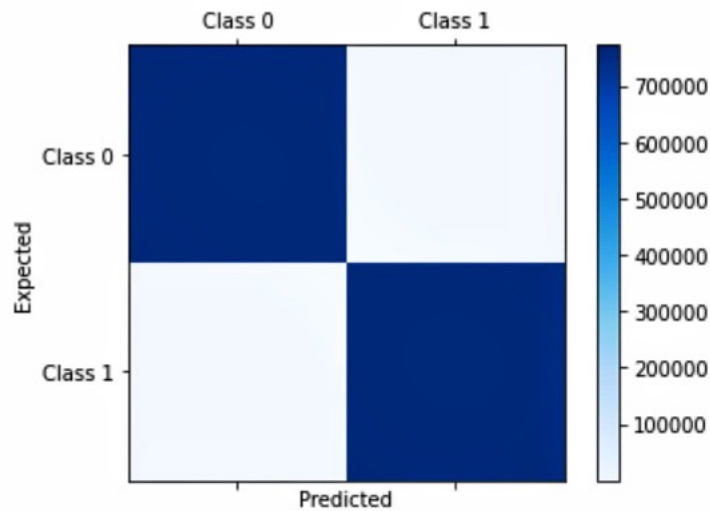


Figure 7.1: *The confusion matrix chart of the k-NN model*

Here, in the K-NN model it shows the high number of true positive and true negative cases with 771554 and 769703 respectively. Besides, a low number of false negative and false positive with 55973 and 1293 respectively.

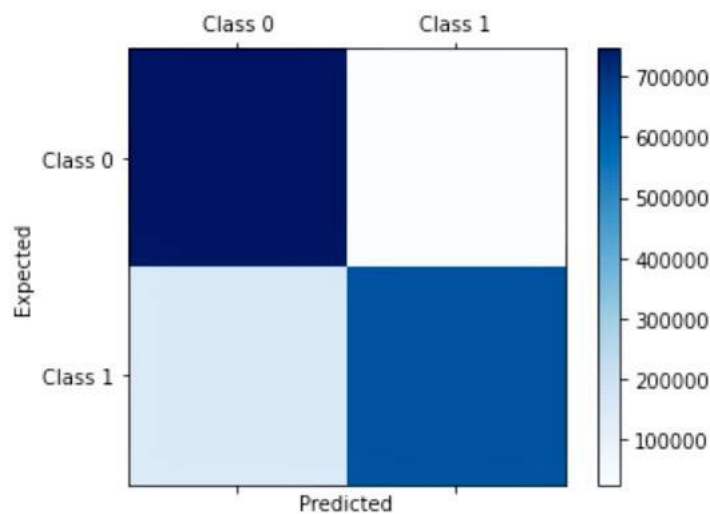


Figure 7.2: *The confusion matrix chart of the SVM model*

Here, in the SVM model it shows the high number of true positive and true negative cases with 746148 and 622784 respectively. Besides, a low number of false negative and false positive with 146942 and 25426 respectively.

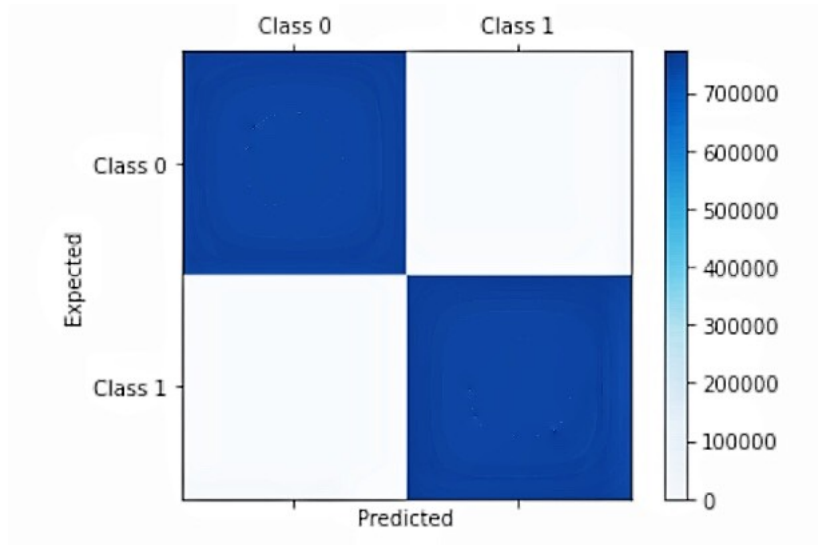


Figure 7.3: *The confusion matrix chart of the Random Forest model*

Here, in the Random Forest model it shows the high number of true positive and true negative cases with 771574 and 769721 respectively. Besides, a low number of false negative and false positive with 46955 and 1206 respectively.

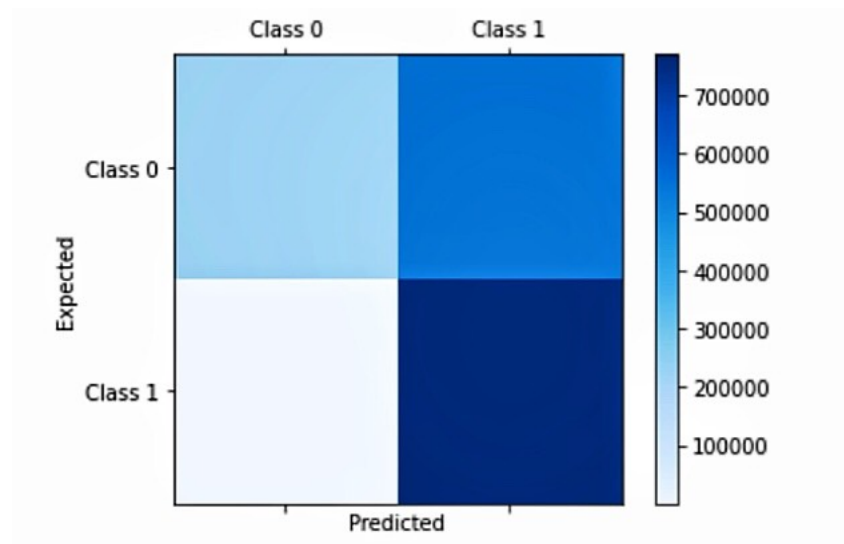


Figure 7.4: *The confusion matrix chart of the Naive Bayes model*

Here, in the Naive Bayes model it shows the high number of true negative and false positive cases with 769461 and 549156 respectively. Besides, a low number of true positive and false negative with 222418 and 265 respectively.



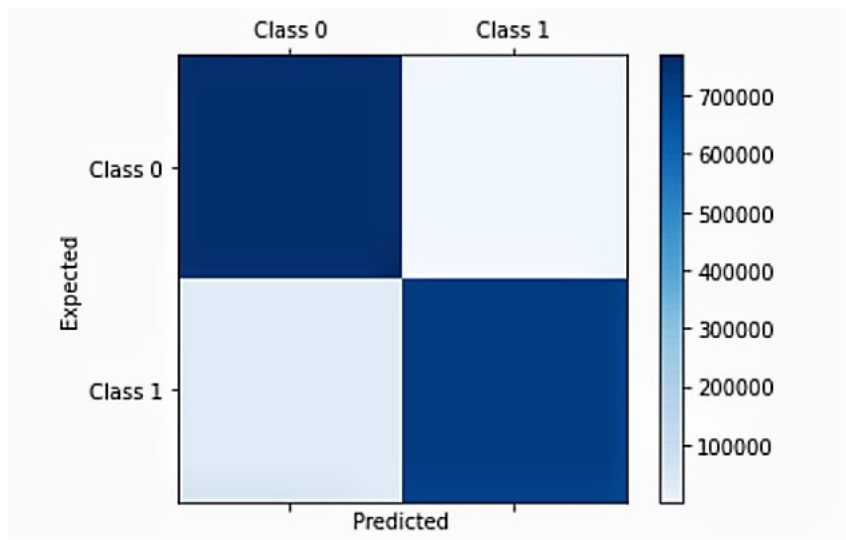


Figure 7.5: *The confusion matrix chart of the Neural Network model*

Here, in the Neural Network model it shows the high number of true positive and true negative cases with 769993 and 719344 respectively. Besides, a low number of false negative and false positive with 50382 and 1581 respectively.

# Chapter 8

## Experimental Results and Analysis

In this part, we analyze the achievement of the Machine Learning models used in this study. As a consequence of studying the experimental findings presented in the tables [] and [], we can conclude that the Random Forest, k-NN, and Neural Network models performed much better than the other two models, which were the SVM and the Naïve Bayes models. The accuracy of the Machine Learning models is indicated in the bar graph in Figure []. Random Forest and k-NN models have reached maximum accuracy of 96.97 and 96.42 percent. Among the other pre-trained Machine Learning models, Neural Network achieved 92.62 percent accuracy, SVM achieved 88.81 percent accuracy, and Naïve Bayes achieved 64.35 percent accuracy, respectively. The Naïve Bayes model, on the other hand, achieved the lowest accuracy of 64.35 percent, which is not able to accomplish our criteria because it assumes that each input variable is independent, which is a very unrealistic assumption in a real-world situation. This is the primary reason why the Naïve Bayes model achieves satisfactory accuracy. The studies show that pre-trained models are quite strong and can be more accurate than untrained models due to the significant learning rate seen during training and the faster convergence experienced during testing. In addition, we see that the outcomes of the Random Forest and k-NN models, as presented in Table [] and Table [], are extremely close to one another. It should be observed that all of the high-performing models have a significant number of layers in their design, but the two underperforming models, namely the SVM and the Naïve Bayes model, have a relatively small number of layers in their architecture.

Table 8.1: PERFORMANCE COMPARISON IN TERMS OF ACCURACY, PRECISION RECALL AND F1 SCORE

Models	Accuracy	Precision	Recall	F1
k-NN	96.42%	98.88%	95.39%	96.58%
Random Forest	96.97%	93.89%	99.31%	96.71%
Neural Network	92.62%	90.93%	94.63%	93.55%
SVM	88.81%	86.64%	91.49%	86.94%
Naïve Bayes	64.35%	63.82%	60.46%	65.76%

Accuracy is the ratio of total correctly predicted cases and the total number of cases. Here in the following table, K-NN has an accuracy of 96.42% followed by the random forest classifier accomplished with the highest accuracy of 96.97%. Then, the neural network with an accuracy of 92.62%. SVM perform poorly with an accuracy of only 88.81%. and naïve Bayes with the lowest accuracy of only 64.35%. Besides, precision is the ratio of total correctly predicted positive cases and the total predicted positive cases. Here, we can see the K-NN and the Random Forest classifier has the precision of respectively 98.88% and 93.89%. On the other hand, naïve Bayes has the lowest precision of 63.82%. Additionally, recall is the ratio of total correctly predicted positive cases and total positive cases. Similarly, here also random forest classifier shows the highest recall value of 99.31% followed by K-NN with the value of 95%. Naïve Bayes shows the lowest recall value of 60.46%. Lastly, the f1 score is the weighted average of precision and recall. Here, the random forest classifier has the highest level with 96.71%. Then, K-NN and NN have the f1 value of 96.58% and 93.55% respectively. Naïve Bayes shows the F1 of 65.76% which is the lowest rather to other classifiers. In short, we can see from the table that RFC accomplished the best performance meanwhile naïve Bayes shows very poor performance.

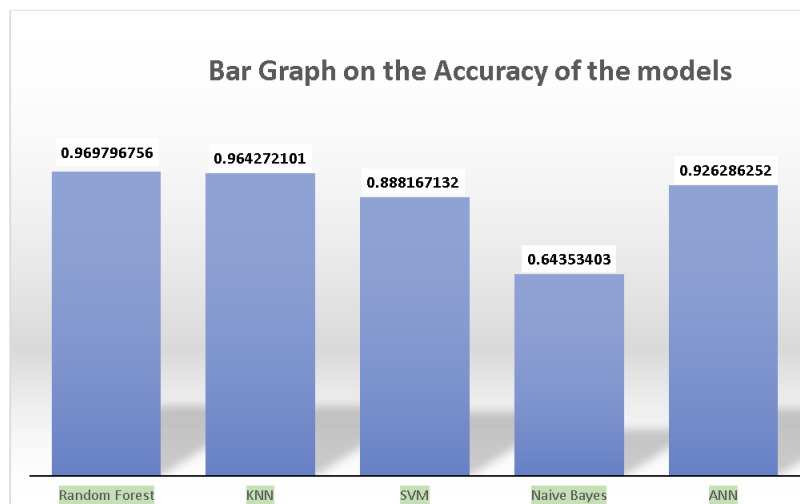


Figure 8.1: Bar Graph on the Accuracy of the ML Model

Table 8.2: PERFORMANCE COMPARISON IN TERMS OF AUC AND CONFUSION MATRIX

		CONFUSION MATRIX			
MODELS	AUC Score	TP	FP	FN	TN
k-NN	96.42%	771554	1293	55973	769703
Random Forest	96.97%	771574	1206	46955	769721
Neural Network	92.62%	769993	1581	50382	719344
SVM	88.80%	746148	25426	146942	622784
Naïve Bayes	64.39%	222418	549156	265	769461

The area under the curve (AUC) is the measurement of the ability of a classifier to differentiate among the classes. Here in the following table we can see that RFC has the highest AUC score of 96.97% followed by k-NN 96.42%, NN 92.62% and SVM and naïve bayes have a poor score with 88.80% and 64.39% respectively.

# Chapter 9

## Conclusion

In our research we examined IoT related security problems as part of the major study topic by detecting DDoS attack patterns. In order to accomplish this goal, we conduct extensive study on DDoS attack patterns in the data we have collected. Furthermore, we do a detailed study on Machine Learning algorithms' capabilities that are mainly the key attributes for originating an attack detection model. Therefore, in order to identify DDoS attacks, we advised that a deep search can be implemented in patterns correlating labeled malicious data and then we can identify authentic attacks with the help of machine learning classifiers. we marked the normal and DDoS data based on our obtained dataset. Then, we applied supervised machine learning techniques thus we detect DDoS using five classification algorithms as k-NN, SVM, Random Forest, Naïve Bayes, and Neural Network. The Neural Network algorithm we chose is called Artificial Neural Network which gave us some interesting results. To carried out our experiments in order to evaluate the performance of the algorithms against the dataset, we used ROC curves and confusion matrix. The results show Random Forest, k-NN and Neural Network (with 96.97%, 96.42% and 92.62% accuracy) perform with high accuracy while SVM (88.81% accuracy) and Naïve Bayes (64.35% accuracy). We focused only on a few attributes in this thesis work because these are the most important features to detect an attack. According to our understanding, researching and testing with key aspects, the results may provide understanding on how to identify all forms of attacks in the dataset. In our findings, very few studies have used the supervised machine learning approach to identify Internet of Things based distributed denial of service (DDoS) attacks. In future, we hope to find more accuracy in DDoS attack detection of IoT devices and try new methods to strengthen our findings.

# Bibliography

- [1] Steward, J. (2021, December 5). 21+ Internet of Things Statistics, Facts & Trends for 2022. Findstack. Retrieved January 16, 2022, from <https://findstack.com/internet-of-things-statistics/>
- [2] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3), 1-58.
- [3] Johnes, S. (2021, April 2). 10 IoT Trends to Disrupt the Tech World in 2021. Iotforall. Retrieved January 16, 2022, from [https://www.iotforall.com/10-iot-trends-to-disrupt-in-2021?fbclid=IwAR1BWMBGI1g5N\\_dXXwNqCtLxLMzjr3KWOKWM9MsNsx9On05YGdP6M3KBhQ](https://www.iotforall.com/10-iot-trends-to-disrupt-in-2021?fbclid=IwAR1BWMBGI1g5N_dXXwNqCtLxLMzjr3KWOKWM9MsNsx9On05YGdP6M3KBhQ).
- [4] Wang, K., Dong, J., Wang, Y., & Yin, H. (2019). Securing data with blockchain and AI. *Ieee Access*, 7, 77981-77989.
- [5] What is a distributed denial-of-service (DDoS) attack? (n.d.). Cloudflare. Retrieved January 16, 2022, from <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>
- [6] Zeadally, S., Adi, E., Baig, Z., & Khan, I. A. (2020). Harnessing artificial intelligence capabilities to improve cybersecurity. *Ieee Access*, 8, 23817-23837.
- [7] Pramana, M. I. W., Purwanto, Y., & Suratman, F. Y. (2015, August). DDoS detection using modified K-means clustering with chain initialization over landmark window. In *2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)* (pp. 7-11). IEEE.
- [8] Abdullah, M. S., Zainal, A., Maarof, M. A., & Kassim, M. N. (2018, November). Cyber-attack features for detecting cyber threat incidents from online news. In *2018 Cyber Resilience Conference (CRC)* (pp. 1-4). IEEE.
- [9] Abeshu, A., & Chilamkurti, N. (2018). Deep learning: The frontier for distributed attack detection in fog-to-things computing. *IEEE Communications Magazine*, 56(2), 169-175.
- [10] Ullah, F., Naeem, H., Jabbar, S., Khalid, S., Latif, M. A., Al-Turjman, F., & Mostarda, L. (2019). Cyber security threats detection in internet of things using deep learning approach. *IEEE Access*, 7, 124379-124389.
- [11] Doshi, R., Apthorpe, N., & Feamster, N. (2018, May). Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)* (pp. 29-35). IEEE.

- [12] Lin, C. T., Wu, S. L., & Lee, M. L. (2017, August). Cyber attack and defense on industry control systems. In 2017 IEEE Conference on Dependable and Secure Computing (pp. 524-526). IEEE.
- [13] Farooq, H. M., & Otaibi, N. M. (2018, March). Optimal machine learning algorithms for cyber threat detection. In 2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim) (pp. 32-37). IEEE.
- [14] Rao, A., Carreón, N., Lysecky, R., & Rozenblit, J. (2017). Probabilistic threat detection for risk management in cyber-physical medical systems. *IEEE Software*, 35(1), 38-43.
- [15] Nguyen, H. V., & Choi, Y. (2010). Proactive detection of DDoS attacks utilizing k-NN classifier in an anti-DDoS framework. *International Journal of Electrical, Computer, and Systems Engineering*, 4(4), 247-252.
- [16] Schölkopf, B., Smola, A. J., & Bach, F. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.
- [17] What is a distributed denial-of-service (DDoS) attack? (n.d.). Cloudflare. Retrieved January 16, 2022, from <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>
- [18] Ajagekar, S. K., & Jadhav, V. (2018, May). Automated approach for DDOS attacks detection based on naive Bayes multinomial classifier. In 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 1-5). IEEE.
- [19] Nicholson, C. Evaluation metrics for machine learning-accuracy, precision, recall, and f1 defined, 2019.
- [20] Ahanger, T. A. (2017, March). An effective approach of detecting DDoS using artificial neural networks. In 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET) (pp. 707-711). IEEE.