Real-Time DDoS Detection in Software-Defined Networks Using Machine Learning

by

Kadir Hasan
20101332
Kaji Sajjad Hossain
20101321
GM Mohaiminuzzaman Apurbo
20301100
MD Zubairul Islam
20101322
Md Shakibul Alam
20301286

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

Department of Computer Science and Engineering
Brac University
May 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/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:

Kadir Hasan
20101332

Kaji Sajjad Hossain
20101321

GM Mohaiminuzzaman Apurbo
20301100

Md Shakibul Alam
20301286

MD Zubairul Islam
20101322

Approval

The thesis/project titled "Real-Time DDoS Detection in Software-Defined Networks Using Machine Learning" submitted by

1. Kadir Hasan (20101332)
2. Kaji Sajjad Hossain (20101321)
3. GM Mohaiminuzzaman Apurbo (20301100)
4. MD Zubairul Islam (20101322)
5. Md Shakibul Alam (20301286)

Of Spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May, 2024.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor-1:
(Member)

Mr. Md Faisal Ahmed
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor-2:
(Member)

Dr. Jannatun Noor Mukta
Assistant Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

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

Head of Department:
(Chair)

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

Abstract

As the landscape of the digital world keeps changing and getting more advanced, so do the sophistication and complexities of cyber threats. Distributed Denial of Service (DDoS) attacks have become a major threat to network security. Additionally, in software defined networks (SDN), the structure uses a controller to track down the network flow. In this research, we worked with a traditional static dataset, "CICIoT2023" in order to detect DDoS attacks on IoT devices with an efficient approach by applying effective feature engineering using Random Forest and PCA, followed by comparing various machine learning models including Random Forest, KNN, Decision Tree (DT), Logistic Regression (LR) and Naive Bayes. Using only 3 key features out of 47, the research shows that Random Forest selection method gives better accuracy for most of the ML models. Among those ML models, Decision Tree shows 99.97% accuracy with optimal model complexity. Our study also focused on constructing a network topology using Mininet simulation tool and Ryu controller in a SDN environment, which further complies with DDoS detection in real-time networks. Therefore, our research is not only focusing on the efficiency of the traditional approach but also on generating real-time networks to detect DDoS attacks simultaneously.

Keywords: DDoS Attacks, Detection, SDN, Cyber Threats, Machine Learning, Real Time, Network, Mininet, Ryu, CICIoT2023

Dedication

We would like to dedicate our thesis report to our parents who have always supported us in all aspects of our work. Without their support we may not be able to complete this. We also want to dedicate this paper to our supervisor and co-supervisors who helped us throughout the years. Special gratitude to our friends and close ones.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.

Secondly, to our supervisor Dr. Muhammad Iqbal Hossain sir who relentlessly supported, mentored and guided us through a challenging topic. We were able to overcome the obstacles because of his unwavering support and constant feedback.

Thirdly, we would like to take this chance to thank our co-supervisors, Mr. Md Faisal Ahmed sir and Dr. Jannatun Noor Mukta mam for their continuous support and help throughout the years. Without their contribution, it would be hard and impossible for us to complete our work.

Table of Contents

List of Figures

List of Tables

Nomenclature

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

ACK Acknowledgement

CIC Canadian Institute for Cybersecurity

CSV Comma-separated Values

$DDoS$ Distributed Denial-of-Service

DoS Denial-of-Service

DT Decision Tree

$ICMP$ Internet Control Message Protocol

IoT Internet of Things

IP Internet Protocol

KNN K-Nearest Neighbor

LR Logistic Regression

NB Naive Bayes

PCA Principal Component Analysis

RF Random Forest

SDN Software-defined Network

$SMOTE$ Synthetic Minority Oversampling Technique

SYN Synchronize

TCP Transmission Control Protocol

UDP User Datagram Protocol

VM Virtual Machine

Chapter 1

Introduction

In today's interconnected and digitalized world, the SDN network have become one of the attractive targets for malicious actors seeking to exploit vulnerabilities in cyberspace. The cyberspace term refers to a global digitized sharing environment throughout the world. This includes a vast number of components that play a significant role in data sharing and transfer [9]. In a SDN environment, it's easy and efficient to simulate a DDoS attack and compare it with the real time approach to distinguishing whether a network is benign or malicious. The increasing complexity of these networks may come up with various cyber attacks like DDoS/DoS attacks, reconnaissance, web-based attacks, brute-force, spoofing and Mirai botnets which also stand out as huge threats to tackle. This study aims to detect DDoS attacks by evaluating some machine learning models while using the traditional data on IoT devices with more efficiency and accuracy, and also simulate a real time DDoS attack in a software-defined network and precisely detect benign and malicious traffic using KNN architecture. Furthermore, this research proposes some future work on the idea of mitigating DDoS attacks in such a situation.

1.1 Cyber Security and DDoS

A distributed denial-of-service (DDoS) attack uses a large number of compromised computers (also called botnets) to flood a target's network with a large volume of traffic. It is a different approach to DoS (denial of service) attacks which specifically target only a single user/network. On the other hand, DDoS is a type of DoS attack that uses multiple networks to generate traffic which is also sent to the target system. DDoS is more complicated than DoS as it is more difficult to defend against attackers as the traffic comes from multiple sources [25]. The first time human beings speculated about a DDoS attack was in 1999 when a computer at the University of Minnesota had huge traffic congestion caused by 114 different computer networks which were called Trin00. It worked as a botnet sending millions of data packets within a short period of time to prevent legitimate data requests to the network which resulted in two days of computer service blockage [7]. While applying DDoS, the attackers use a huge amount of node traffic to compromise their target and later consume the data from the network. This results in destroying the whole server infrastructure and prevents users from using their network. DDoS attacks have two techniques to implement: reflection and amplification. There are also three types of DDoS attacks, volume based, protocol based and application based. All these

types have their own distinct target attributes to capture the network system of their target [25]. Let's talk about the categories first.

- **Volume Based:** These types of DDoS attacks aim to render a system inaccessible by overloading the traffic by executing exploitation. Mostly, it used the UDP protocol to take advantage of the unnecessary growth of packet size. In addition to that, servers increase the bandwidth of the network which creates much more traffic, and the huge volume of packets causes inaccessibility. [23]

- **Protocol Based:** Protocol based attacks mostly focus on the server side rather than bandwidth like the volume based DDoS attack. In this case, attackers send unnecessary packets to the target destination and eat up all the available resources. As it is executed in the network layer, attackers try to exhaust different hardware like CPU, RAM and device storage. A crucial element of resource depletion attacks is taking advantage of TCP communication protocol characteristics. The most common variant of this attack is called a TCP SYN Flood, and it uses the TCP protocol to establish a three-way connection in order to consume all of the capacity allocated for connection management (backlog). The target server then has to wait for the client's permission to finish the connection establishment stage, which never occurs. Lastly, it lessens the backlog, which stops new connections from being made [23]. In this research, we do a protocol-based DDoS attack data analysis.

- **Application Based:** Attacks targeting the application layer aim to exploit vulnerabilities in a service or application that may cause instability and prevent authorized users from accessing the system. One such attack on the application layer is Slowloris. In order to maintain its effectiveness, the Slowloris attack uses the least amount of bandwidth feasible, which makes it invisible to monitoring systems that are triggered by abnormalities in network traffic [23].

1.2 DDoS Attack Cases

Nowadays, DDoS attacks are now becoming one of the major concerns in networking security. Most of the companies and farms have their own servers and networking systems which are easily vulnerable to different malicious attacks, most importantly DoS/DDoS attacks. Throughout the year, there are some significant events of DDoS attack in cyber security history that have grabbed headlines in newspapers over the years.

The AWS (Amazon Web Server) attack in February 2020 was one of the biggest attacks in cyber security history [41]. The attacker sent almost 2.3 terabits of packets per second to their server which weighed down their bandwidth and temporarily shut down all the hosts that were connected to the server. The attackers used TCP SYN and UDP packets to flood the system. They used connection-less lightweight directory access protocol (CLDAP) to access the network. Though AWS shield mitigated the largest DDoS attack ever very precisely.
Before the AWS attack, the GitHub server attack was regarded as the biggest DDoS attack at the time [45]. In 2018, a group of hackers found a vulnerability in the

cached system (also known as memcached) where they sent 1.3Tbps of information to the server. They didn't use any zombie or bot hosts but instead they took advantage of memcached to amplify their bandwidth high enough to jam the system's traffic. After 20 minutes, GitHub's strong DDoS protection mitigated the attack successfully.
Another biggest DDoS attack was recorded back in 2023, when Google's cloud server received almost 398 million requests per second (rps) from the attackers. On their website, they compared the attack as the two minute attack generated much more requests than the total number of Wikipedia articles viewed by users in the entire month of September, 2023 [44]. Their investigation found the attack was initiated using the 'Rapid Reset' technique to generate such a large number of requests. Figure 1.1 shows the prediction of DDoS attack until 2023 by a CISCO report.

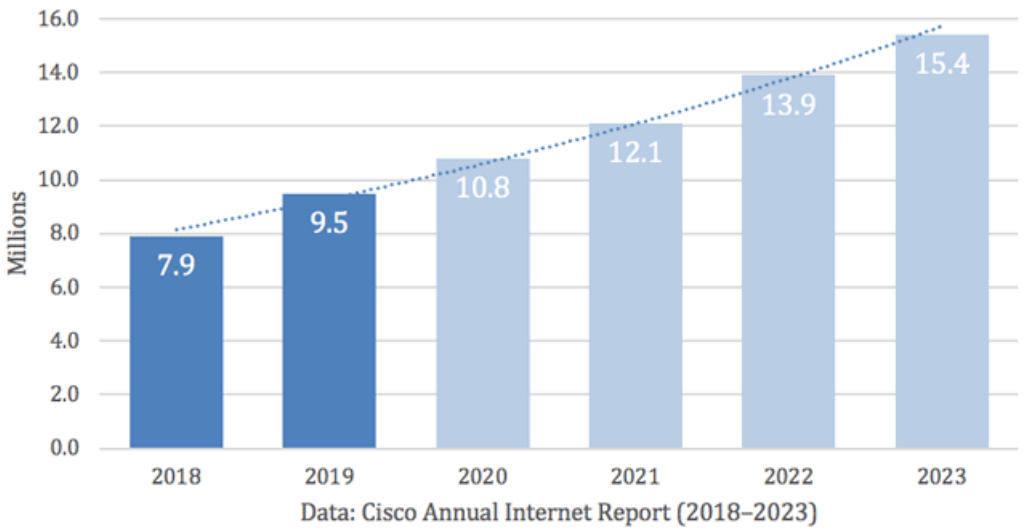


Figure 1.1: Cisco's Analysis of DDoS Total Attack History and Predictions

Although the largest DDoS attacks have been mitigated successfully throughout the years, it is becoming a major concern about the future and the rapid growth of malicious attackers. The technological advancement is one of the major reasons for increasing the number of DDoS attacks. To tackle DDoS attacks, an individual must detect the attack as soon as possible. DDoS detection is the process that distinguishes DDoS attacks from normal cyber attacks. There are many methods that are applied today that are becoming more and more sophisticated. To detect the attack, an individual must gain productive knowledge about the whole networking system. Then different types of machine learning and deep learning needed to be applied. Different deep learning models can be applied to detect DDoS among other cyber attacks. In summary, hybrid machine and deep learning models can be easily implemented to detect any DDoS attack very precisely.

1.3 DDoS Attack in SDN Network

In this real time world, software based simulation has a rapid growth in all sectors of technology management. Software defined network (SDN) is a simulation based

approach which uses controllers and advanced interfaces to simulate a network traffic to identify the fundamental characteristics of the network of a system. To ensure security, many developers use SDN based controllers to detect abnormal traffic since it have a large scale of network system [42]. Nowadays, the SDN environment also has the biggest risk of DDoS attacks as they interrupt the flow of network services by mitigating the network service and bandwidth. The primary concept of SDN is to remove the primary network service with the help of a centralized controller [5].

A basic SDN controller consists of three different layers: data, control and application layers. The third layer known as application layer, is responsible for handling different applications, hosts, IDS (intrusion detection system), IPS (intrusion prevention system) and other firewall applications. The controller is used for managing traffic control flow and other customizations like packet control (drop and flow management), flow forwarding, etc. In our paper, we initialized the RYU controller to capture the packet flow. Control and data layer communicates with each other with bounded APIs such as OpenFlow and NetConf which are called northbound interface [3]. The last layer, which is data (also called infrastructure) layer, is submerged with different networking devices such as router, switch and other access points. Different virtual switches such as vSwitch, Pica8, Indigo, Nettle, OpenFlow are included in this layer [1]. In the control layer, a controller has the main functionality to flow the network simulator in a SDN environment. The communication between the control and data layer has an interface called southern interface. Figure 1.2 [6] shown below perfectly defines the SDN architecture and necessary layers.

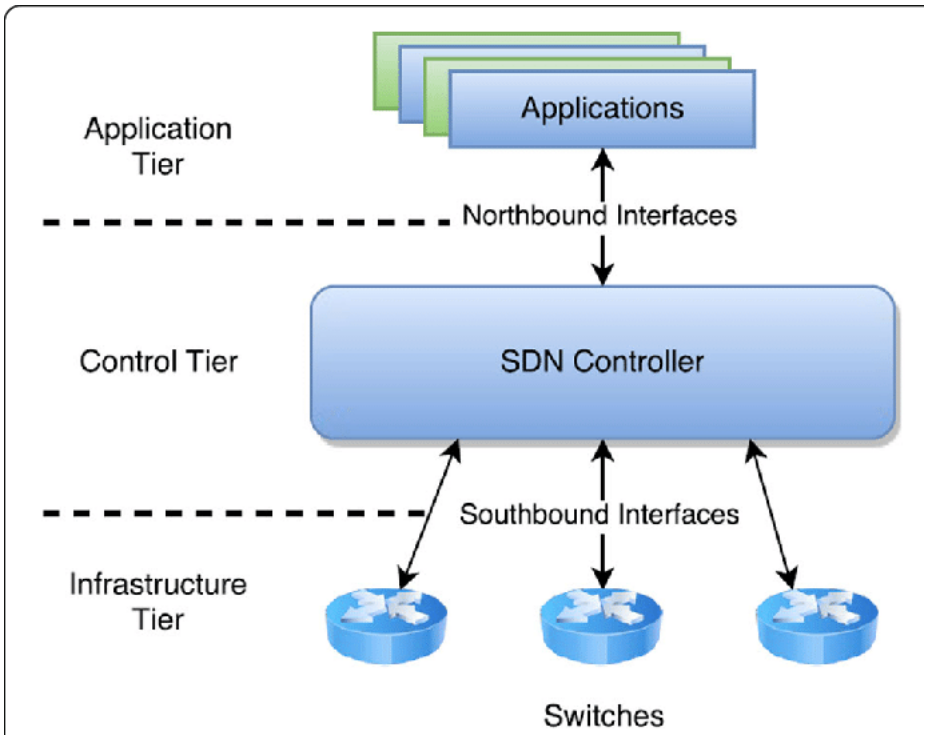


Figure 1.2: SDN Architecture

The SDN controller is generally known as the brain of SDN where all the decision making is implied according to the simulation. DDoS attacks are typically generated by UDP, ICMP and TCP packets where a large number of packets are sent to the

server to slow down the network traffic. In our paper, we used Mininet to create a virtual network and RYU as a SDN controller. Mininet is an open source software that generally emulates a SDN network. It contains different topologies that consist of several numbers of hosts, servers, switches and routers. They all connected through a central controller which generated network traffic in the simulation [2].

As a controller, RYU is a famous open source controller which is programmed in python and supports OpenFlow protocol of a SDN network. RYU can develop different network applications including OFconfig and the open virtual switch database (OVSDB) that can modify different rules of protocols in a network flow [16]. The structure in fig 1.3 shows how the RYU controller is structured and how different layers can contribute to create communications between users and OpenFlow switches. RYU is much more developed in terms of supporting higher versions of OpenFlow protocols like OFF 1.5 and also supporting extensions like Nicira. That's why we used this controller with the simulation tool Mininet to establish our own network topology in our work. Figure 1.3 shows the basic architecture of a RYU controller with different layers.

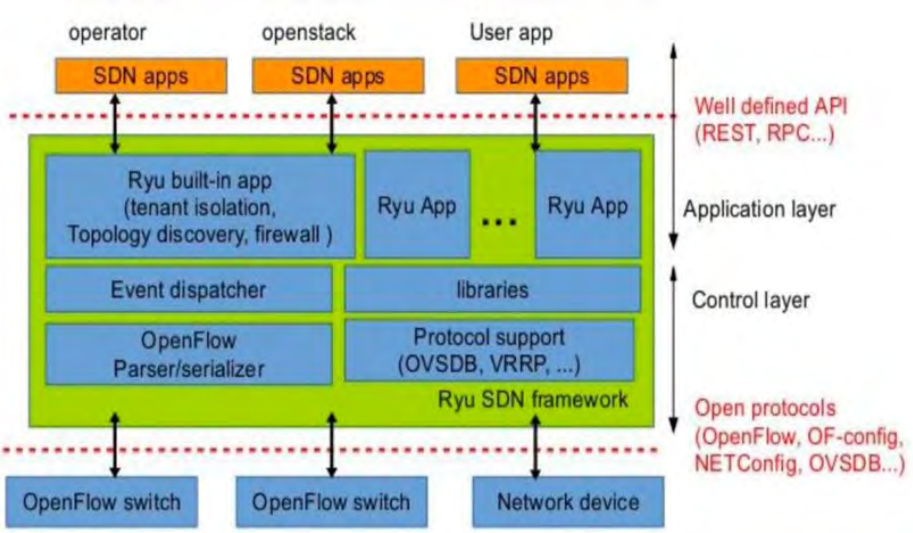


Figure 1.3: The Architecture of RYU Controller

1.4 Research Problem

Just as the digitalization of technology keeps carrying forward, the rate of threats towards technological devices also tends to keep boosting. Previously, denial of service (DoS) attacks played a malicious role in the world of cybercrime, but gradually, DDoS also began to enlarge and develop on large scales. One of the biggest risks in SDN architecture is the DDoS attack. So detecting DDoS attacks using the traditional method is not sophisticated as the network can dynamically change in real time situation. In some of the traditional works, there are many drawbacks to using static network flow datasets. In order to solve these problems, we proposed a real time method to integrate the same findings to detect DDoS attacks in traditional ways.

In Norton, they broke down the categories of DDoS attacks into two different types based on the intended outcomes of the hackers:

- Flooding attack: This refers to an attempt of applying overwhelming flood of data to take down a server. An ICMP or ping flood occurs when an attacker or hacker sends data packets to affect an entire group of connected computers, thereby bringing down that misconfigured network.
- Crashing attack: The rare DDoS variant involves blocking a compromised system with bugs that affect weaknesses in the targeted infrastructure. Once these unpatched vulnerabilities are exploited, the system will crash.

In some of the research, like [38], they used a traditional dataset and proposed a hybrid model to detect DDoS effectively but in most cases, it was much more complicated to integrate the dataset into real time approach. As a result, static network flow is generated which could be a major drawback. In our work, we not only proposed an efficient method of DDoS detection in traditional datasets but also a system where real time DDoS attacks can be generated.

Some studies, like [4] mostly focused on SDN environment to detect DDoS attacks but didn't provide any comprehensive network statistics. Although SDN generated data can sometimes lower their scability and effectiveness, it also sometimes does not offer enhanced security features. So, focusing only on a SDN environment could lead to inappropriate data capture and sometimes even less accuracy in defining malicious attacks in real time scenarios. We evaluated the idea of capturing both methods in our paper to ensure the scalability, effectiveness, security and accuracy of detecting DDoS using both traditional and real time approach.

In another research [13] proposed a hybrid dataset by combining a traditional and SDN dataset by using some applications. In most cases, combining different datasets could result in dropping the data quality. For example, difference in data accuracy, reliability and completeness can affect the novelity of a dataset. Furthermore, data cleaning might be needed to solve incoherence and data complications. In addition, there could be other issues like data duplication, nullability and semantic misalignment. That's why our research doesn't focus on combining traditional datasets with SDN-based simulated datasets to ensure accuracy and data validity.

In short, this research solely focuses on two different procedures of data validation to ensure scalability. Previous research and studies either focused on traditional datasets or SDN environments to detect DDoS attacks. But we managedto evaluate them as efficient and accurate as possible. We use several data pre-processing methods to filter out inconvenient features from the traditional dataset and use some machine learning models like Random Forest, Decision Tree, Naive Bayes, Logistic Regression and KNN to detect DDoS attacks more efficiently. We also proposed a simulated system where DDoS can also be detected in real time scenarios.

1.5 Research Objective

This research aims to detect DDoS attacks using two different types of data capture. Firstly, we observe the efficiency and effectiveness of a traditional dataset where DDoS attacks can be effectively detected by data processing and secondly, we evaluate a network simulator to execute real time network flow capture to improve our findings. The objectives of this research are:

1. Deeply understand how the datasets work.
2. Process the traditional dataset to get a more efficient and cost-effective structure.
3. Evaluate various machine learning models to ensure the accuracy of detecting DDoS attacks in the modified dataset.
4. Construct a network topology in SDN environment to execute DDoS attack in real time scenario.
5. Detect DDoS attacks by using KNN in SDN environment.
6. Explore the efficiency and drawbacks of each dataset and find the relative comparison and usability of traditional and real time networks.

Chapter 2

Literature Review

Dandotiya et al. [38] proposed a deep learning method for detecting DDoS attacks in the SDN environment. Their methodology involved beginning the pre-processing of the CICDDoS2019 dataset; which consisted of the benign and various DDoS attack scenarios. They splitted the dataset into training and testing sets to facilitate model training and evaluation. The authors proposed a hybrid deep learning algorithm which is Stacked LSTM+CNN architecture. Their architecture automatically extracted significant DDoS features from the mentioned dataset and classified those features. Their stacked model achieved an accuracy of 99.76%, a precision of 99.64%, and a perfect recall of 100%, outperforming other solo models namely CNN, GRU, LSTM, and Bidirectional LSTM. However, their proposed architecture has some limitations. Firstly, their proposed architecture may not cover all possible attack scenarios. Secondly, their architecture achieved an accuracy in recall of 100%, which will result in overfitting; they may fail to analyze new data. Furthermore, the authors did not mention any future proposals regarding the mitigation of DDoS attacks.

Tahirou et al. [36] detected and mitigated DDoS attacks in SDN using supervised machine learning techniques. The authors proposed their system within the SDN controller, Ryu. They applied Naive Bayes (NB), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) to classify traffic as normal or abnormal based on the CIC-DDoS2019 dataset. Their methods involved extracting IP address and port numbers from the mentioned dataset and blocking the IP addresses or port numbers of attacking hosts by using system software named Snort; thus mitigating the attacks. The accuracies obtained by the authors for Naive Bayes, K-Nearest Neighbors and Support Vector Machine are 80.65%, 92.25% and 98.5% respectively. However, the effectiveness of the proposed mitigation methods heavily depends on the accuracy of the ML classifiers, which may vary in dynamic network environments. Furthermore, the dataset they used may not represent real world network scenarios.

Liu et al. [28] proposed a novel approach to detect and mitigate DDoS attacks in the SDN networking environment on the basis of feature engineering and machine learning algorithms. They pre-processed the CSE-CIC-IDS2018 dataset and applied an improved binary grey wolf optimization algorithm (GWO algorithm) for feature selection, reducing the dataset from 79 features to 26 features. These features are trained and tested by Random Forest (RF), Support Vector Machine (SVM), K-

Nearest Neighbor (k-NN), Decision Tree, and XGBoost. The results indicate that the Random Forest classifier performed the best by achieving an accuracy of 99.13%, precision of 98.43%, recall of 99.92%, and F1 score of 99.13%. In comparison, XGBoost had the highest accuracy on the original dataset at 96.9%, but RF showed superior performance overall after feature extraction. This RF classifier then monitored network traffic and identified DDoS attack traffic and blocked the incoming packets. However, the proposed method is only applicable for large datasets, which can be a limitation in resource-constrained environments.

Wang et al. [37] proposed a defense framework named CC-Guard for detecting and mitigating DDoS attacks on SDN controllers. Their framework's system design consisted of four modules: the attack detection trigger module, the switch migration module, the anomaly detection module, and the mitigation module. The attack detection module is initiated when the ratio of incoming packet messages per unit exceeds a certain threshold value compared to the controller's processing capacity. The switch migration module prevented the SDN controller from overloading by migrating switches from an overloaded controller domain to a less utilized one; ensuring the continuity of the normal operation of SDN. The anomaly detection module was utilized to classify between the normal activity and DDoS attacks. This module is involved in applying deep learning models, namely CNN-GRU-Attention model, to which they achieved an accuracy of 99.63%, precision of 99.77%, recall of 99.58%, and F1-score of 99.67%. Finally, the authors proposed a cross-domain method to mitigate the attack. They used the SDN controller's east–west interfaces for cross-domain traceability and employed the OpenFlow protocol to issue flow constraints for blocking the abnormal switch ports. Nevertheless, the proposed framework's applied Information Entropy in classifying DDoS attacks from benign activity may not be effective. Moreover, the framework is applicable only for simplified and small topologies.

Jemal et al.[27] proposed the CNN model, a deep learning technique for detection of DoS and DDoS attacks in IoT devices. The paper's methodology comprises the use of several layers in the CNN model, including the embedding layer, convolutional layer, max-pooling layer, fully connected layer, and softmax layer. These layers collaborate to identify and classify various forms of attacks. The authors used the DOS-TCP, DOS-UDP, DDOS-TCP, and DDOS-UDP attacks of the Bot-IoT dataset to train and test their proposed model. The authors achieved an impressive accuracy rate of 99.92% after training the CNN model. Moreover, the authors also showcased the superiority of the CNN model in detecting DoS and DDoS attacks on IoT by comparing the performance of the CNN model with other existing approaches on the same dataset based on ASCII codes. The paper lacks a full explanation of the suggested CNN model's potential limitations and obstacles, such as scalability, adaptability to developing attack tactics.

Sadhwani et al. [34] proposed a model that detects DDoS attacks on the network created among the lightweight IoT devices based on feature extraction and application of many machine learning algorithms. They detected the DDoS attacks on lightweight IoT devices by pre-processing the two datasets- BOT-IOT and TON-IOT datasets. Their pre-processing includes data standardization using SMOTE

approach and feature extraction using ExtraTreeClassifier (ETC) to extract 15 most relevant features from each of the mentioned datasets. Those features are then classified as normal attack and DDoS attack respectively using binary and multiple class classifiers namely logistic regression, random forest, naïve bayes, artificial neural network (ANN), and k-nearest neighbor (KNN). The performances of these machine learning classifiers were distinctive. For binary classification, all models achieved 100% accuracy but for multiple-class classification, only random forest achieved 100% accuracy, while ANN achieved 99% accuracy. The authors observed that in the BOT-IOT dataset, Naive Bayes acquired the highest accuracy for both binary and multiple-class classification; whereas in the TON-IOT dataset, Naive Bayes was the best model for binary classification and Random Forest was the best model for multiple-class classification. However, despite applying many ML models, the authors may have faced overfitting issues as most of the models achieved 100% accuracy. Additionally, the Naive Bayes classifier has performed poorly in multiple-class classification for the TON-IOT dataset.

Ahmim et al. [22] proposed a hybrid approach for detecting Distributed Denial of Service (DDoS) attacks in the Internet of Things (IoT) context. Their purpose is to accurately classify benign traffic as well as all sorts of DDoS attacks, including the most identical ones. The hybrid model was developed by the authors by merging deep learning models namely CNN, LSTM, Deep Autoencoder, and DNN. The authors used the CICDDoS dataset (2019) for training their model. During the preprocessing of the dataset, the authors designed their proposed hybrid model by decomposing it into two models: the global model and the sub- model. The first level of the global model consists of sub-neural networks namely CNN, LSTM, Deep Autoencoder and DNN that have been trained by self-created algorithms containing instances and classes. The second level takes the output of the first level together with the starting data. On the other hand, the one level sub-model contains 2D CNNs, LSTM, and an autoencoder, and the other level contains the output of the first level - the Dense layers. After training, their proposed hybrid model achieved an accuracy of 98.98%. Firstly, the proposed model's training period is considered to be very long. Secondly, the authors did not use precision or recall, and the F1 score may provide a more nuanced evaluation of the model's capabilities.

Gupta et al. [26] proposed an approach that involved Mininet simulation tool, sFlow simulation tool and some lightweight algorithms for real time detection and mitigation of DDoS attacks in SDN topologies. The authors created an SDN environment with OpenFlow switches and an OpenDaylight (ODL) controller using the Mininet tool. The sFlow tool is the core tool of their proposed model. Using that, they captured numerous traffic flows and exported those for further analysis. Subsequently, the authors dynamically set some threshold values in order to distinguish between the benign attack and DDoS attack and mitigate the DDoS attacks. They set the threshold values for mitigation by determining some parameters namely packet average, bytes average and standard deviation. The system then created the new data flow by modifying the attacked flow using those derived parameters from attack characteristics and thus updating the relevant switch with the new flow. The results demonstrated that their proposed system was able to handle up to 70,000 packets per second during attack scenarios. However, their system required high CPU usage,

peaking at 100% which resulted in complexity.

Raju et al. [32] addressed the growing threat of IoT-Botnet attacks by improving Network Intrusion Detection (NID) through the use of Machine Learning algorithms. Considering the increasing number of IoT devices and evolving cyber threats, they explored the application of ML algorithms to detect and mitigate IoT-Botnet attacks. The authors used Decision Tree Classifier, Random Forest Classifier, Logistics Regression, KNN and Gaussian Naive Bayes. They used the CIC-IoT (2023) dataset for pre-processing. They used the Kali Linux tool to convert the files of the dataset to CSV format. Their pre-processing consisted of missing values, categorical encoding, and feature scaling. The mentioned ML algorithms are trained and tested on the pre-processed data. The experimental results show that the authors selected 20 important features out of 47 features in the dataset. After training, the achieved accuracies are 97.19% (Decision Tree Classifier), 99.11% (Random Forest Classifier), 98.22% (KNN); but the remaining algorithms achieved very low accuracies. Regarding the feature selection, the authors did not explain why they consider certain features as important. Moreover, they did not clearly explain the simulated attacks.

Chouhan et al. [19] proposed feature extraction and classification to detect DDoS attacks in Ryu controller based SDN. Their approach involved the extraction of seven key features from their own generated dataset. They generated their own dataset by creating attacks using the hping3 tool on their created SDN topology. The extracted features are trained and tested by applying machine learning algorithms namely Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbor (KNN), eXtreme Gradient Boosting (XGBoost), and Naive Bayes (NB). The application of these machine learning techniques helped them to detect DDoS attacks in real time. Among the techniques, the SVM classifier achieved highest accuracies across all the evaluated metrics; an accuracy of 99.398%, precision of 99.413%, recall of 99.397%, false alarm rate (FAR) of 0.718%, area under the curve (AUC) of 0.995, and F1 score of 99.4%. Even though achieving better accuracy, SVM takes more testing time than the other mentioned techniques.

Sharma et al. [35] improved security in smart cities by presenting a fog computing-based architecture for detecting and forecasting DDoS attacks. They explored and evaluated the performance of various machine learning methods, namely Random Forest, Naive Bayes, and Decision Tree on the BoT-IoT (2021) dataset. The dataset is trained using Random Forest (RF), Naive Bayes (NB), and Decision Tree (DT). Pre-processing is performed on the dataset to manage missing values, convert data types, and pick pertinent features. The models' performance is then assessed using metrics, namely accuracy, precision, recall, and F1-score. The accuracies of RF, NB and DT are 91%, 71% and 91% respectively; showing NB has the least accuracy. However, the authors presented a limited discussion of how fog computing improves security in the context of smart cities.

Aslam et al.[18] proposed a framework named AMLSDM (Adaptive Machine Learning based DDoS detection and Mitigation) to detect and mitigate DDoS attacks in SDN-enabled IoT environments using adaptive machine learning techniques. The framework is composed of real-time network traffic classification and the mitigation

of DDoS attacks. They created their network topology using Pox and Floodlight SDN controllers. For pre-processing, they collected some network statistics and converted those to csv format, creating a primary dataset of their own. Then they extracted the features, namely Number of Packets, Size of Bytes, Number of source IPs, and destination IPs. Upon those features they employed ML techniques namely Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF) and k-Nearest Neighbor (k-NN) for classification. The highest accuracies achieved in terms of precision, recall and F1-score were 97%, 98% and 97% respectively. In the mitigation phase, the authors mitigated the classified attacks by enabling the open flow switches to drop the incoming packets and switching the network resources to the legitimate network hosts. As a result, they dynamically reallocated network resources to maintain service availability for legitimate users. However, the proposed framework may face challenges in scaling to very large networks with highly dynamic traffic patterns.

Li et al. [21] proposed an approach named DoSGuard to attack and mitigate DoS attacks in SDN networks. DoSGuard comprised three core components: a monitor, a detector, and a mitigator. The monitor maintained the mapping relationship between switches and hosts to quickly identify network anomalies, while the detector analyzed OpenFlow message and flow features to detect attacks. The mitigator then blocked malicious traffic by installing flow rules at the switch level. The authors applied ML algorithms, particularly Support Vector Machine (SVM), which achieved high accuracies in precision (96.77%), recall (100%), and F1 score (98.34%), outperforming other algorithms tested. Furthermore, DoSGuard demonstrated superior detection effectiveness compared to existing methods such as entropy-based detection and DoSDefender. However, the approach had some shortcomings. Firstly, it only blocked compromised hosts rather than specific flows. Secondly, DoSGuard focused on protecting the control plane and data plane, neglecting threats on the application plane and northbound interfaces.

Ashfaq et al. [17] focused on the classification of normal and DDoS attacks in IoT networks using machine learning techniques. The authors used Wireshark and KDD Cup datasets for pre-processing; they divided the data set based on source and destination IP addresses, and the accepted number of requests. The authors applied Logistics Regression and Decision Tree techniques to classify the DDoS attacks. Depending on the packet size and protocol level of the data of the mentioned datasets, the authors predicted the types of DDoS attacks on IoT networks by running Logistics Regression and Decision Tree algorithms. The authors predicted a total nine types of DDoS that can breach on IoT networks. Now to classify them under three different features namely volume based attack, protocol based attack and application layer attack; the authors used Random Forest Classifier technique. The Random Forest Classifier technique also helped them in avoiding overfitting. The experimental results show that the Logistics Regression model achieves an accuracy rate of 99.99% and 99.8% for Wireshark and KDD Cup datasets respectively. While the Decision Tree model achieves an accuracy rate of 99.999% and 99.7% for Wireshark and KDD Cup datasets respectively. The authors compared their performance level with KNN and SVM models; these models achieved below or around 98% accuracy. However, while the paper discusses accuracy and confusion matrices as performance

measurements, it does not go into detail about other important metrics such as precision, recall, or F1 score. Moreover, the authors did not mention the detailed information about the data packet size of the datasets.

Gaur et al. [20] detected the constant and expanding danger of DDoS attacks on IoT devices using DL technique. The authors used CICDDoS (2019) dataset as their goal is to improve the detection model's classification performance by employing a RNN based LSTM model. The authors used a multi-layer LSTM architecture to identify DDoS attacks. After pre-processing the dataset, they trained the LSTM model on the time series dataset. The LSTM model using its specified parameters investigated three scenarios: binary, ungrouped and grouped multiclass classification. The binary scenario classified whether or not an attack happened. In the ungrouped multiclass, each category of DDoS has been tested for accuracy. The maximum accuracy acquired after using the 2-layer LSTM model was 98.76%. Finally, in the third scenario the imbalanced class labels have been classified into four labels. These labels defined a harmless threat and the types of detected DDoS attacks which were UDP, SYN, NetBIOS, LDAP, and MSSQL. In contrast, the authors didn't present a detailed comparison of their proposed model with other models, instead they just showed that in simple tabular form.

Kousar et al. [13] proposed a detection system that detected DDoS attacks in SDN networks in which the Decision Tree algorithm was the most applicable one. During pre-processing, they initially used CIC-DDoS 2019 dataset and extracted 18 relevant features from raw 84 features using CICFlow meter technique. The authors then applied three machine learning algorithms: Naïve Bayes, SVM, and Decision Tree on the extracted features for binary and multiple-class classification. On the other hand, the authors generated their own dataset on the SDN by creating attack traffic by using the Hping3 tool and collecting those traffic data using an OpenFlow switch and storing in pcap files. Then they again applied those mentioned machine learning models for classification. After application of the models, the authors found that during binary classification the Decision Tree showed highest accuracy rates of 99.93% for the CIC-DDoS 2019 dataset and 96.1% for the SDN dataset (generated dataset); during multiple-class classification, the accuracies were 99% and 93% respectively.. But other models they trained on both the datasets were showing overfitting issues. Therefore, in order to prove the validity of their dataset, they created a hybrid dataset which comprised SDN dataset as training dataset and CIC-DDoS dataset as testing dataset. During binary classification, Decision Tree achieved 94.6% and 99% during multiple-class classification for the hybrid dataset. However, the author could have used some deep learning techniques for feature extraction instead of CICFlow meter that could have result in better detection.

Tawfiq et al. [15] developed a middleware system to mitigate the cyber attacks on homes that are connected via IoT devices. The authors intend to address the growing security difficulties confronting smart homes that rely on IoT devices that are exposed to various sorts of cyber-attacks. The authors integrated SDN and DL techniques (CNN and LSTM) to enable automatic security against cyber-attacks on IoT networks. DL techniques are used to analyze and filter IoT data flow, identifying and responding to threat activity, while SDN was integrated with those models

for allowing dynamic and centralized network control. For low-cost implementation, the proposed system architecture makes use of the Raspberry Pi and the Zodiac-Fx SDN open flow switch. The authors used the IoTID20 dataset and preprocessed it to reduce the number of features. After training the models, they achieved 98.3% accuracy (highest) during binary classification and 86.1% accuracy (highest) during multi-classification. But the authors claimed that even with advanced machine learning approaches, the system has been unable to detect new types of IoT assaults, posing a threat to IoT networks.

Doshi et al. [11] developed an effective IDS system capable of identifying and mitigating DDoS attacks on IoT devices. They proposed two strategies: Detection Strategy and Mitigation Strategy. The detection strategy involved a non-parametric statistical anomaly detection algorithm known as the Online Discrepancy Test (ODIT) was implemented which is a decision-making system that utilizes accumulated evidence rather than relying on individual data points to inform its decisions. Therefore, the algorithm was able to detect such low rate nature DDoS attacks. The mitigation strategy involved an hierarchical structure in which all the local nodes are connected to a central entity through a topology. The nodes share their data to the central entity, which then targets that specific attacked node and makes precise blocking of that node. The experimental results show that the proposed ODIT-based IDS system detected real-time HTTP flooding attacks, including various attacks at onset of 80 seconds. The authors tested a hierarchy of 10 nodes, each monitoring 100 IoT devices. However, this hierarchy might not detect all possible variations of DDoS attacks. Moreover, the proposed algorithm requires accurate parameterization of the normal data which is challenging as the features of IoT devices in the real world vary greatly.

Wang et al. [10] proposed a comprehensive framework to address security concerns, particularly by detecting DDoS attacks in Internet of Things (IoT) applications. The proposed framework combines SD-IoT controllers, switches, and IoT devices to provide centralized control and real-time monitoring. The real time monitoring of the integrated controller of the framework provides a number of data packets at a fixed interval. The authors then used a deep learning detection technique (CNN) based on time series data that uses parameters such as average packets per flow, average bytes per flow, and source IP address entropy. The authors also evaluated the performance level of CNN by utilizing machine learning techniques namely DNN, SVM and KNN. The experimental results show that the CNN model outperforms the applied machine learning methods. The CNN model gives 98.7% accuracy while the DNN, SVM and KNN models achieve below that. As a result, the paper contributes to the SDN-IoT security landscape by presenting a careful integration of SDN and IoT, as well as a comprehensive solution to DDoS attack detection. While extracting the features of DDoS attack, the paper did not provide sufficient justification for their selection. Moreover, their paper lacks detailed explanation of the metrics used to evaluate the performance of the proposed model.

Gautam et al. [8] proposed a layered based architecture aiming to mitigate the sniffing and spoofing DDoS attacks in the Ryu and Pox based SDN networks. Layer 2 security was involved protecting the data packets that are transmitting within the

SDN structure itself , while Layer 3 security was involved securing communication between the network segments controlled by SDN. The authors created an attack by directly connecting their hosts to the controller (without the intervention of switch). They used wireshark to capture and analyze network traffic. To detect and block the DDoS affected packets, they implemented a firewall application on SDN controller for packet filtering based on source and destination MAC addresses, source and destination IP addresses, and port numbers. After analysis, the authors found the network based on Ryu controller maintained better performance than Pox controller under high volume attack scenarios. Up to 4000 packets were sent to simulate a flooding attack on Ryu controller and approximately 50% of the flooding traffic was filtered out. The reason behind these is Ryu has its own varieties of applications that can work as reference applications to block the malicious traffic. However, the authors faced processing overhead while filtering packets. Moreover, they carried out the simulations on simple topology in both controllers. Furthermore, the authors didn't thoroughly explain how they used their mentioned layers as module for detection and mitigation of DDoS attacks.

Chapter 3

Datasets

3.1 Data Analysis

The dataset we are using for our research is called CICIoT2023, which is constructed by an extensive topology composed of a good number of IoT devices that are acting as attackers or victims [30]. Figure 3.1 shows the basic experimental setup to generate the dataset framework. The authors collected multiple types of data from 33 different types of attacks and combined them into seven classes against 105 IoT devices. The categories of the attacks have been divided: DDoS, Denial of Service (DoS), Recon, Web-based, Bruteforce, Spoofing, and Mirai. The attacks were simulated by using different tools in Kali Linux. They evaluated multiple ML and DL models to label the classes and distinguish between benign (normal) attacks and cyberattacks. The network traffic of the topology was monitored, and the data was stored as pcap files using multiple networking software programs like Wireshark and Mergecap. The pcap files were also converted to csv by using dpkt tools.

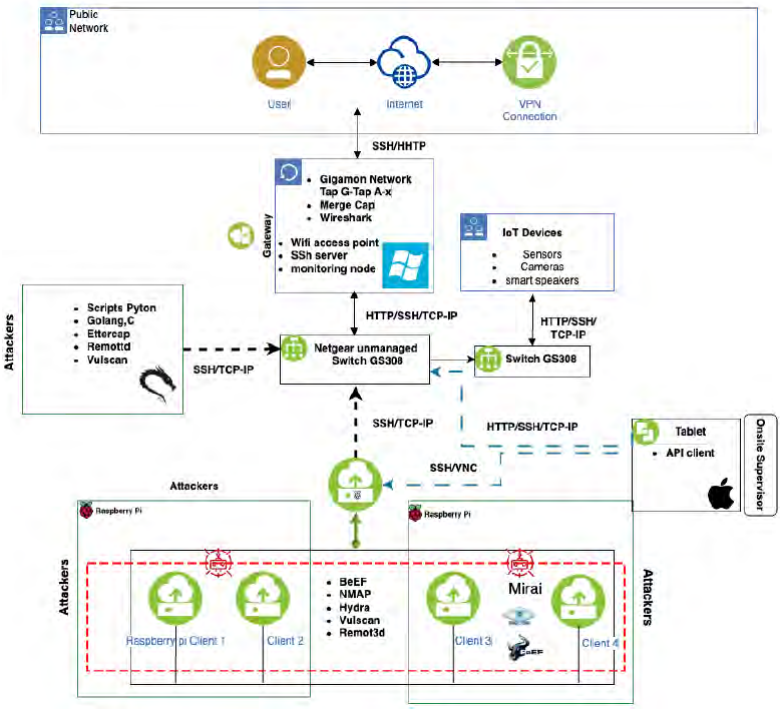


Figure 3.1: Experimental setup for the dataset framework

As there are multiple types of attacks, each attack is performed by different malicious IoT devices. Figure 3.2 shows that the DDoS attack was the most common and dominant attack among all those 7 classes in the IoT network. On the other hand, Figure 3.3 shows number of rows for each scenario.

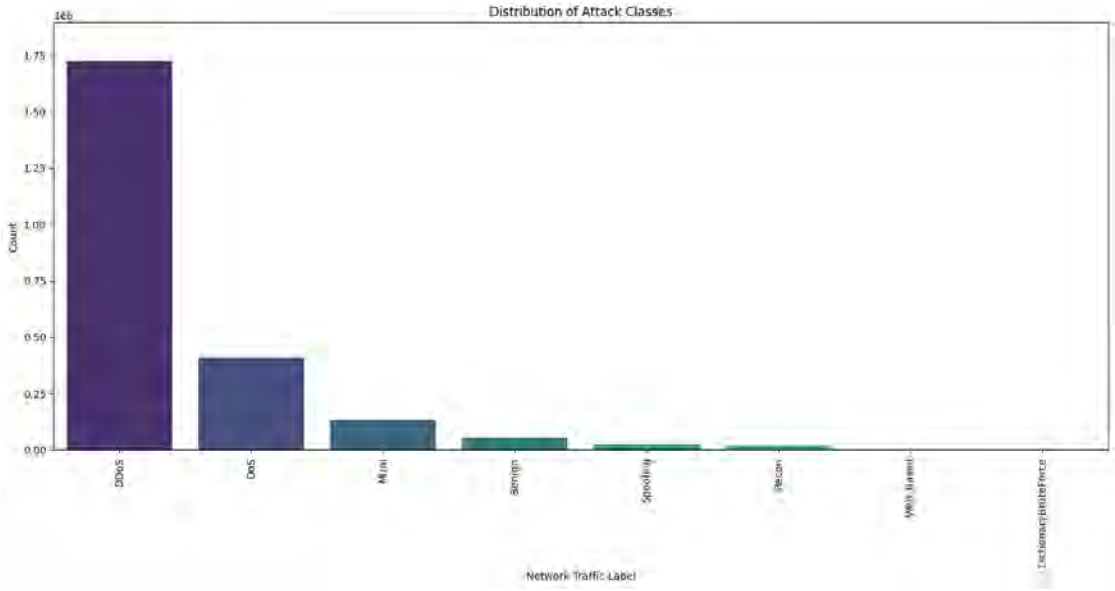


Figure 3.2: Number of rows for each category

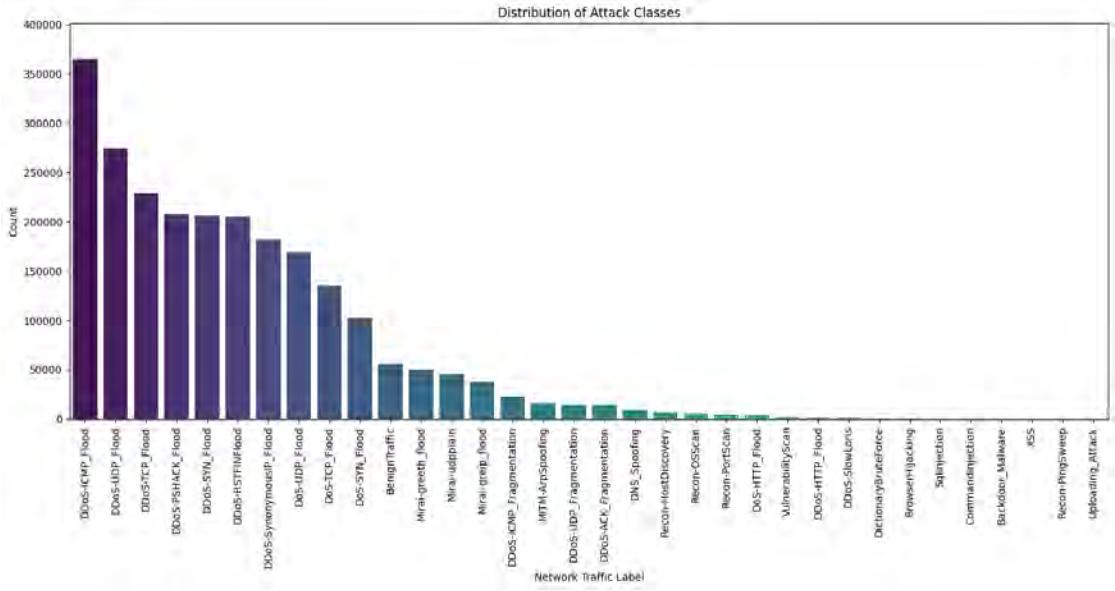


Figure 3.3: Number of rows for each scenario

There are 47 features that are encapsulated between the dataset csv files. These features show different aspects of network traffic and their summaries. Along with this, the features also show various flag values, fundamental attributes and networking protocols. The dataset paper shows the statistical measurement of different features with some derived metrics. Table 3.1 shows all the features and their descriptions alongside.

Feature	Description
flow duration	Duration of the packet's flow
Header Length	Header Length
Protocol Type	e IP, UDP, TCP, IGMP, ICMP, Unknown
Duration	Time-to-Live
Rate	Rate of packet transmission in a flow
Srate	Rate of outbound packets transmission in a flow
Drate	Rate of inbound packets transmission in a flow
fin flag number	Fin flag value
syn flag number	Syn flag value
rst flag number	Rst flag value
psh flag number	Psh flag value
ack flag numbe	Ack flag value
ece flag number	Ece flag value
cwr flag number	Cwr flag value
ack count	Number of ack flags in the same flow
syn count	Number of syn flags in the same flow
fin count	Number of fin flags in the same flow
urg count	Number of urg flags in the same flow
rst count	Number of rst flags in the same flow
HTTP	Indicates if HTTP is used
HTTPS	Indicates if HTTPS is used
DNS	Indicates if DNS is used
Telnet	Indicates if Telnet is used
SMTP	Indicates if SMTP is used
SSH	Indicates if SSH is used
IRC	Indicates if IRC is used
TCP	Indicates if TCP is used
UDP	Indicates if UDP is used
DHCP	Indicates if DHCP is used
ARP	Indicates if ARP is used
ICMP	Indicates if ICMP is used
IPv	Indicates if IP is used
LLC	Indicates if LLC is used
Tot sum	Summation of packets lengths in flow
Min	Minimum packet length in the flow
Max	Maximum packet length in the flow
AVG	Average packet length in the flow
Std	Standard deviation of packet length
Tot size	Packet's length
IAT	Time difference with the previous packet
Number	Number of packets in the flow
Magnitude	Avg. of lengths of in/out packets / 2
Radius	Variance of lengths of in/out packets / 2
Covariance	Covariance of lengths of in/out packets
Variance	Variance of lengths of incoming/outgoing packets
Weight	Number of in packets * Number of out packets

Table 3.1: Summary of the Features of the Dataset

In case of DoS attacks, only one device or Raspberry Pi is used but in case of DDoS, multiple devices are used for the SSH based master client configuration. For both DoS and DDoS, the attacks that are executed are:

1. ACK Fragmentation
2. Slowloris
3. UDP, TCP, ICMP and HTTP Flood
4. RST-FIN Flag Flood
5. PSH-ACK Flag Flood
6. UDP Fragmentation
7. ICMP Fragmentation
8. SYN Flood
9. Synonymous IP Flood

The dataset also evaluated some Machine Learning models that were used in different applications. The performance of different models also measured by some of the parameters like: accuracy, recall, precision and F1 score. They measured the performance of the models by classifying them as malicious or benign.

3.2 Data Pre-processing of CICIoT2023

Data pre-processing is one of the major parts of data analysis and model implementation. The CICIoT2023 and dataset is composed of numerical values that need to be processed in several ways. The dataset has 169 separated csv files from which we have selected 10 files to work with. That will make the workflow much easier, more efficient and more accurate in terms of model selection. We used several data visualization techniques for a better understanding of the dataset and the most effective way to process the data.

3.2.1 Class Selection

There are 34 classes in the dataset. As we are doing our research on DDoS attacks only, we are using only 12 of them. Among them, 11 classes are classified as DDoS attacks and one of them is benign (normal). The number of rows varies within these classes.

3.2.2 Label Encoding

Label encoding is the method of allocating numerical labels to values in categorical data. It is a simple yet efficient way of converting categorical data into numerical data suitable for modeling and analysis. The fundamental concept of label encoding is to provide a unique number to each category in a categorical variable. From sklearn, which is a machine learning library for Python, we have imported label encoder which takes a class label and converts it into a numerical value between 0 and the number of class minus 1. In our dataset, we have applied this technique to 12 classes and indicated the categorical values as integer.

3.2.3 Irrelevant Features Removing by Correlation Matrix

A correlation matrix is a table that shows correlation coefficients between multiple features. It also indicates the strength and direction of their linear relationship. It helps to remove irrelevant features from the dataset and make an impact on model building in data analysis and machine learning. We used Python libraries such as Pandas and Seaborn to make correlations among our features. From the correlation matrix, we find out that 7 features do not correlate with others as their values are constant. We removed those features from our dataset. These features are: 'ece_flag_number', 'cwr_flag_number', 'Telnet', 'SMTP', 'SSH', 'IRC', and 'UDP'. Figure 3.4 shows the correlation matrix for all the features.

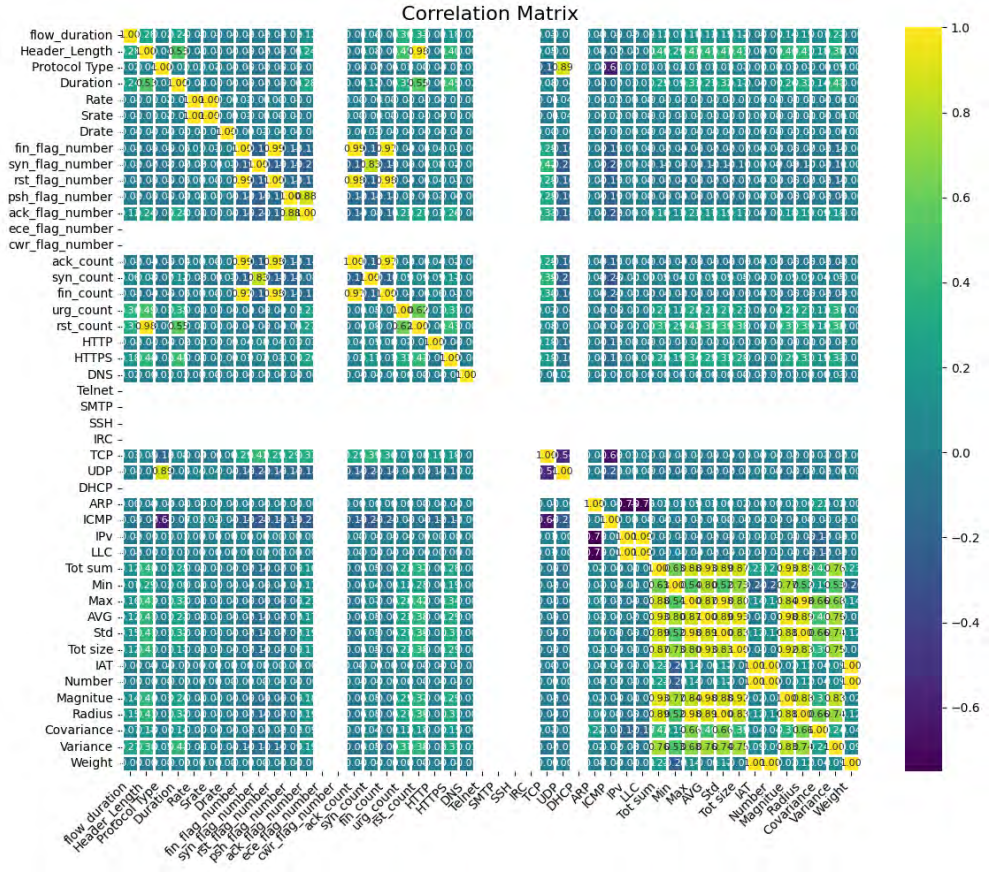


Figure 3.4: Correlation Matrix of All Features

3.2.4 Data Balancing

Our dataset is imbalanced. We applied SMOTE for balancing our dataset using a Python library 'imbalanced-learn'. SMOTE works by generating synthetic samples for the minority class by interpolating between existing minority instances. This helps to balance the dataset and solves issues like diverting biased machine learning models towards the majority class and prevents loss of important data. Figure 3.5 shows the imbalance dataset and 3.6 shows balanced dataset after performing SMOTE.

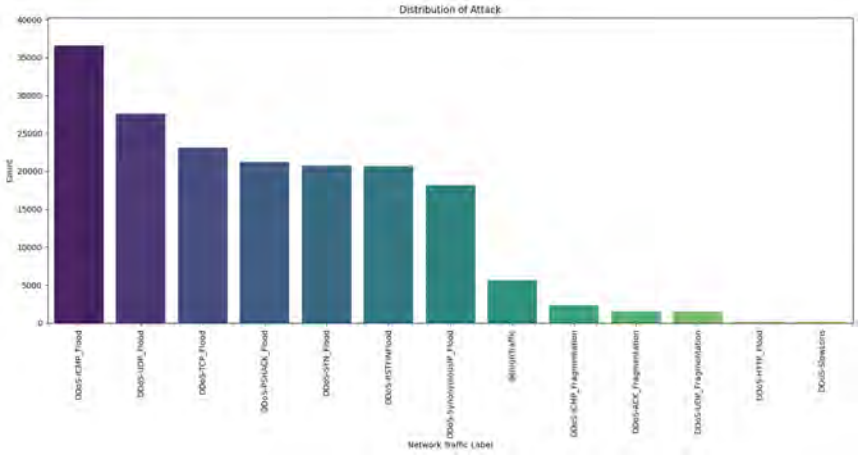


Figure 3.5: Imbalance Dataset

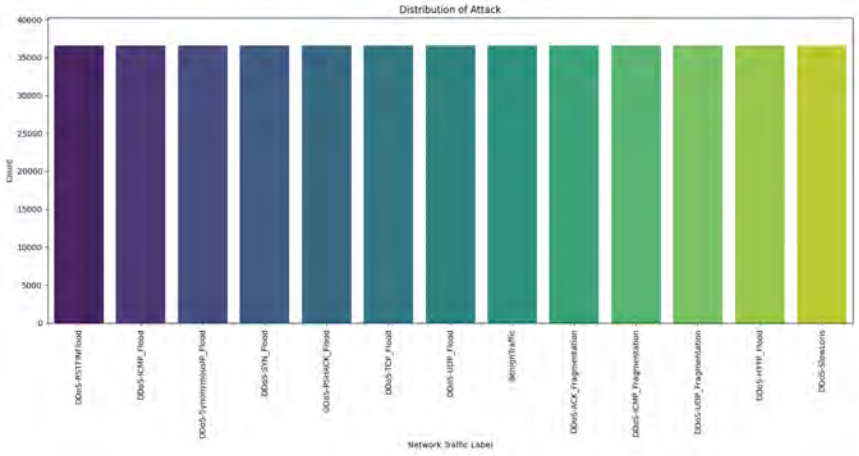


Figure 3.6: Balanced Dataset

3.2.5 Feature Selection

We used the Random Forest model and PCA separately for feature selection on the basis of ascending feature importance score. Random Forest feature importance score helps to figure out which part of the data are most important for the model to make accurate predictions as it works by averaging the decrease in impurity each feature causes across all trees in the forest, indicating how much each feature contributes to the model's predictive power.. This is helpful because it lets us focus

on the essential features, ignore less important ones, and make our model simpler, faster, and easier to understand. Again, we used PCA which works for feature importance selection by transforming the original features into a new set of orthogonal components, ordered by the amount of variance they capture from the data. In that case, features contributing to the components with the highest variance are considered the most important. It is useful as it identifies the most informative features, reducing dimensionality while preserving the essential patterns in the data. We selected 25 features based on Random Forest feature importance and PCA analysis separately and then selected another set of 3 features based on Random Forest feature importance and PCA analysis separately, to compare our model's performance. Figure 3.7 shows the primary and most important features by Random Forest classifier. On the other hand Figure 3.8 shows all the important features if we use the PCA classifier.

If we look closely at Figure 3.9, we see the correlation matrix of both Random Forest and PCA when we are using only 3 features. Again, Figure 3.10 shows the matrix for both cases when we are using 25 features while feature engineering. As per both cases, we can conclude that Random Forest works best in the case of feature selection.

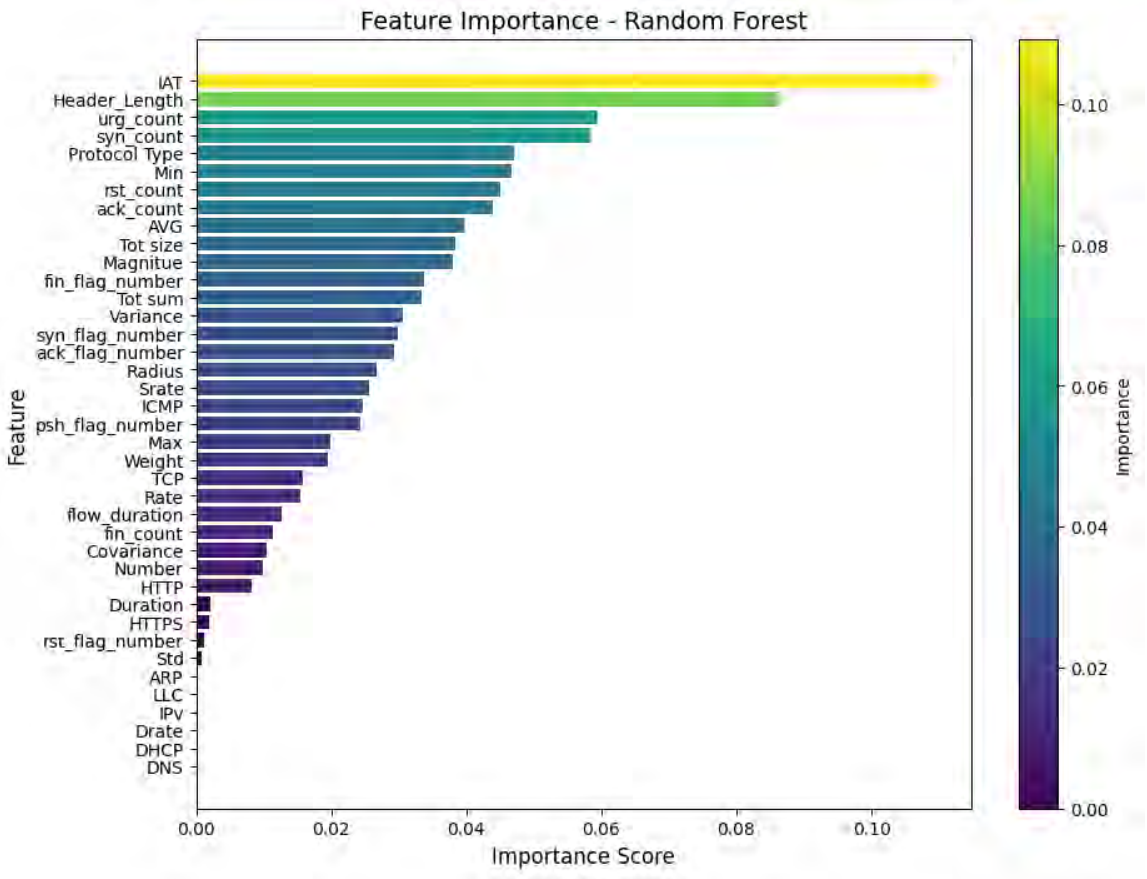


Figure 3.7: Feature Selection Using Random Forest

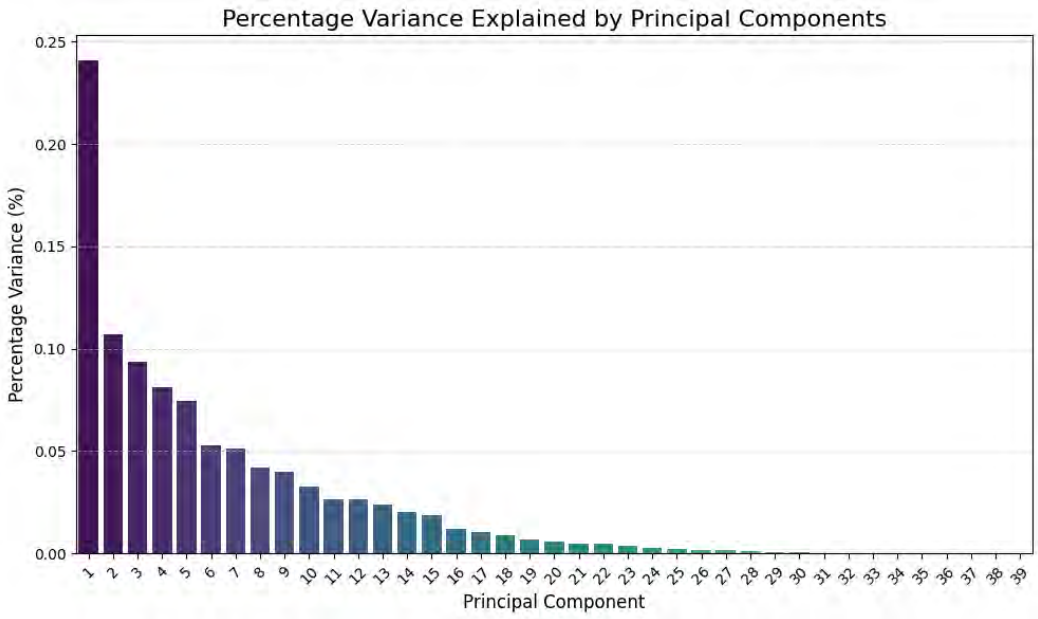
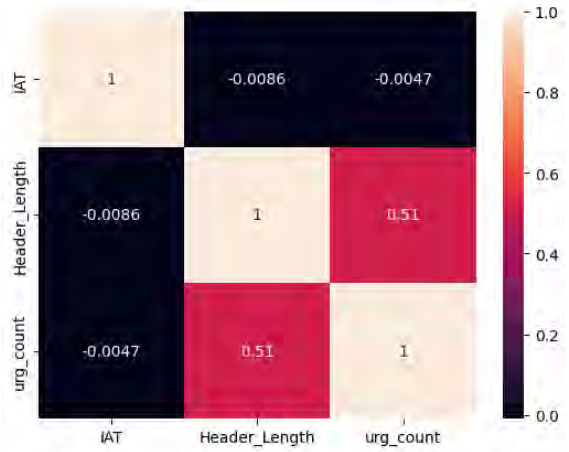
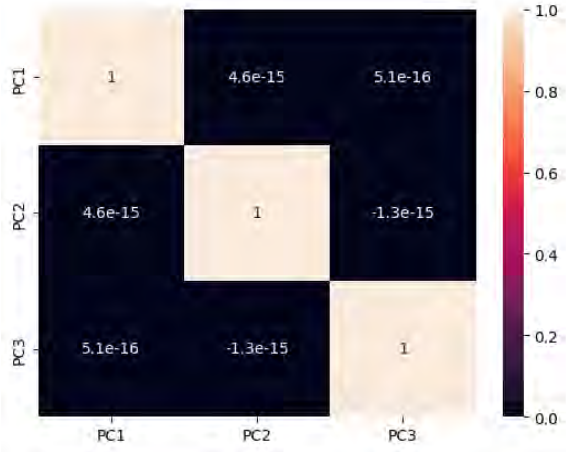


Figure 3.8: Feature Selection Using PCA

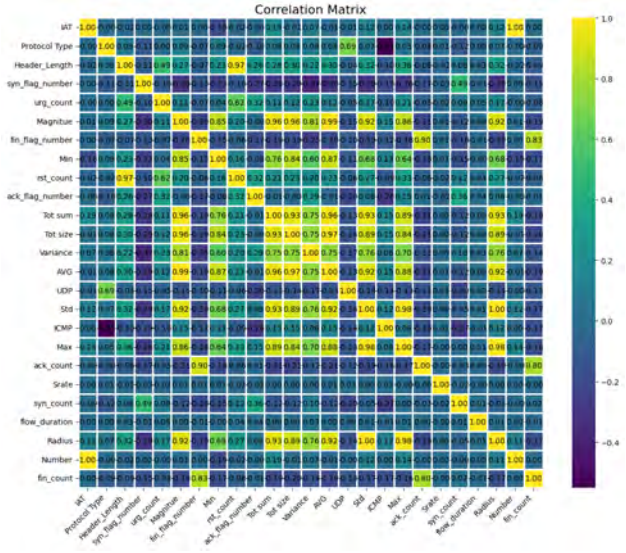


(a) 3 Features for Random Forest

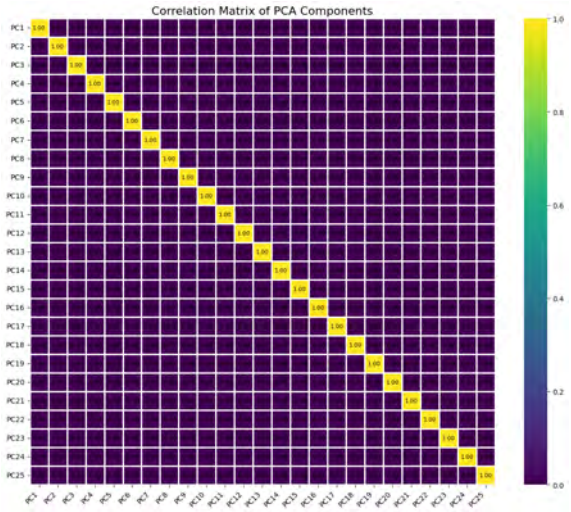


(b) 3 Features for PCA

Figure 3.9: Correlation Matrix Using 3 Features



(a) 25 Features for Random Forest



(b) 25 Features for PCA

Figure 3.10: Correlation Matrix Using 25 Features

3.2.6 Scaling

We applied scaling in data pre-processing to ensure that features with different scales contribute equally to machine learning and deep learning models. Here, we used Standard Scaling by transforming data to have zero mean and unit variance, preventing features with larger scales from dominating the learning process and enhancing model performance.

3.2.7 Data Split

We divided our data into two parts: 80% for training our model, and 20% for testing on how well it learned. We did this using a helpful tool called train_test_split from the sklearn library.

3.3 Benign and DDoS Traffic Generation in SDN Environment

3.3.1 Environment Setup

We used Ubuntu 24.04 LTS as our operating system for its long-term support and stability. We run the OS in an Oracle VM box virtual environment for a flexible and isolated environment. It is a widely used virtualization platform that helps us create a seamless testing environment.

3.3.2 Software Tools

To run a simulation environment we used mininet and RYU controller. Mininet is a powerful Python-scripted simulation tool that allows to perform and test complex topology on a single machine. Using a Linux process, mininet provides a lightweight virtualization simulation process. This makes the tool valuable for experimenting with software-defined networks for testing educational purposes without the need for physical hardware. On the other hand, RYU is a Python-scripted software-defined network (SDN). It provides a manageable framework network that enables users to create a network application that automates the network behavior through programmable interfaces. Lastly, we used the hping3 tool for DDoS attacks. It is a powerful, versatile tool that is used for network testing and others. By using this tool a user can send customs packets and measure responses to the network.

3.3.3 Topology Creation

We create a topology so that we can simulate a network to generate the dataset. The whole environment network is managed by a software-defined network (SDN) mentioned in the figure 3.11 as c0. The controller is connected to 4 switches named s1, s2, s3, and s4. Again, s2 and s3 are connected to two switches, s5 and s6. The topology has a total of 18 hosts, where s1 and s4 are each connected to 4 hosts. s2 and s3 are connected to two hosts. Last but not least, s5 and s6 are connected to a total of six hosts. Three hosts are connected to each network. Later, we generate a Python script to run the code for the topology setup.

In Figure 3.11 we see a network topology created in Mininet which consists of 1 RYU controller, 6 switches and 18 hosts. The fig3.11 is a network diagram that shows several tiers of interconnected nodes arranged hierarchically. The center node at the top, designated "c0," serves as the hierarchy's root. The dashed red lines connecting this root node to multiple intermediary nodes s1, s2, s3, and s4 represent a higher-level link.

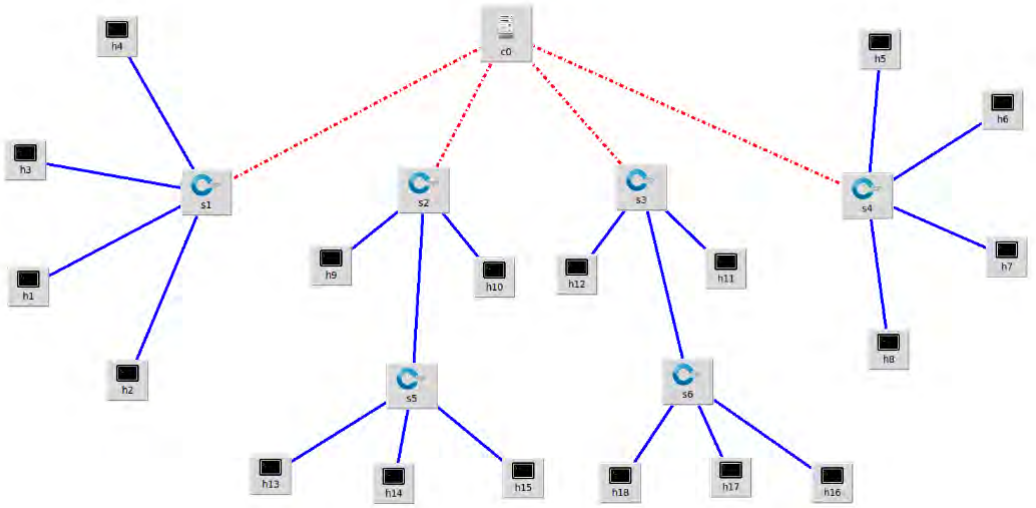


Figure 3.11: Our Constructed Topology

3.3.4 Benign Traffic Generation

For generating benign traffic, in the cmd shell, we activate the RYU controller with a .py script. By RYU SDN controller, we have collected benign traffic by interacting with OpenFlow switches to gather the flow statistics. The SDN controller sets up the monitor switch and a CSV file has been generated to store flow data. The state changes are handled by registering and unregistering switches. The controller collects flow statistics from all switches every ten seconds. After receiving the flow statistics the data was loaded into the CSV file. In that script, it collects the network from the host and writes the stats in a CSV file with 22 columns of features and as it is normal traffic we set the label 0. After that, we run a .py script that pings one host to another host to generate benign traffic. In the script, we first created the topology to run the network simulation. We have then taken the host1 as a server and other hosts as normal hosts. There are a total of 500 iterations to generate the dataset with 10 times random hosts to ping among themselves. Figure 3.12 shows the controller capturing benign traffic in Ryu controller. On the other hand Figure 3.13 shows benign traffic generating while pinging one host to another.

```
(ryu-python3.9-venv) jovial@jovial-VirtualBox:~/controller$ ryu-manager benign_trafic.py
loading app benign_trafic.py
loading app ryu.controller.ofp_handler
instantiating app benign_trafic.py of CollectTrainingStatsApp
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Figure 3.12: Controller to Collect Benign Traffic

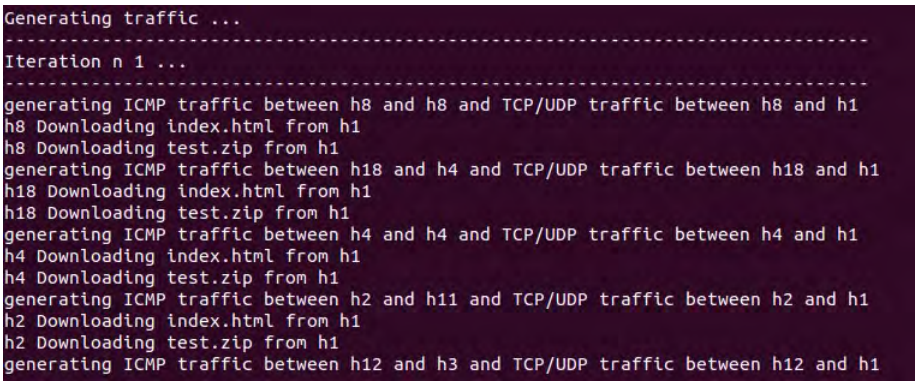


Figure 3.13: Pinging One Host to Another Host For Benign Traffic

3.3.5 DDoS Traffic Generation

In the same way, we did it for DDoS attack traffic. First of all, we ran the controller of DDoS traffic and then we ran the generated scripts with the topology that we have created. In the attack script, we simulated 3 types of attack which are ICMP flood, UDP flood, and TCP flood using the hping3 tool. We used the same topology that we used for the benign traffic. We set all the commands to validate for 20 sec. After that the other command will start for 20 sec and so on. In Figure 3.14 and 3.15 shows collecting and generating DDoS attack by ICMP, UDP and TCP SYN flood.

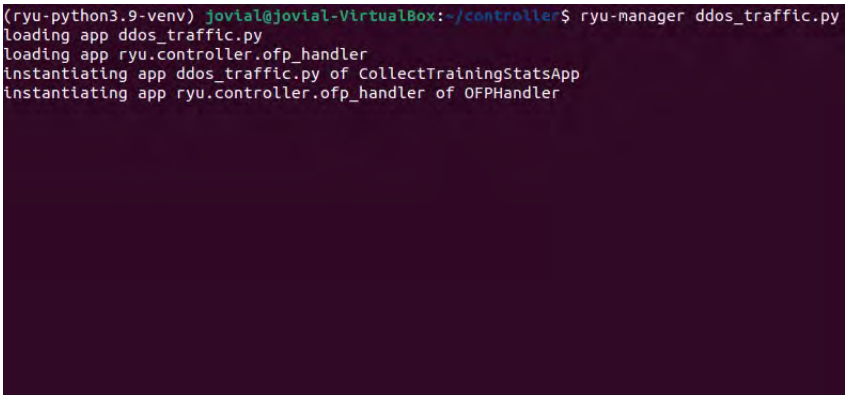


Figure 3.14: Controller to Collect DDoS Traffic

Figure 3.15: Generating DDoS Attack

3.3.6 Data Validation

To check whether our generated traffic is valid or not. We have used Wireshark to see the network flow. In Figure 3.16, the packets are requesting and responding per second. About 10 packets per second were generated in 10-time intervals. For checking the DDoS attacks, when the script ran the command, it gave packets to the host server above 500 packets per second.

In Figure 3.17, we see that the ICMP flood attack is generating and the packet rate rises up to 800 packets per second. Same goes for Figure 3.18 where during UDP flood, the packet rate rises to more than 750 packets per second. In the case of TCP flood, the number of packets per second went up the most. It crossed 1250 packets per second during DDoS attack which can be shown in Figure 3.19. So as per the packet rate count and comparing the number with benign and DDoS traffic, we initially verified that the attacking and benign traffic was genuinely generated in the SDN environment and we also successfully verified our generated data.

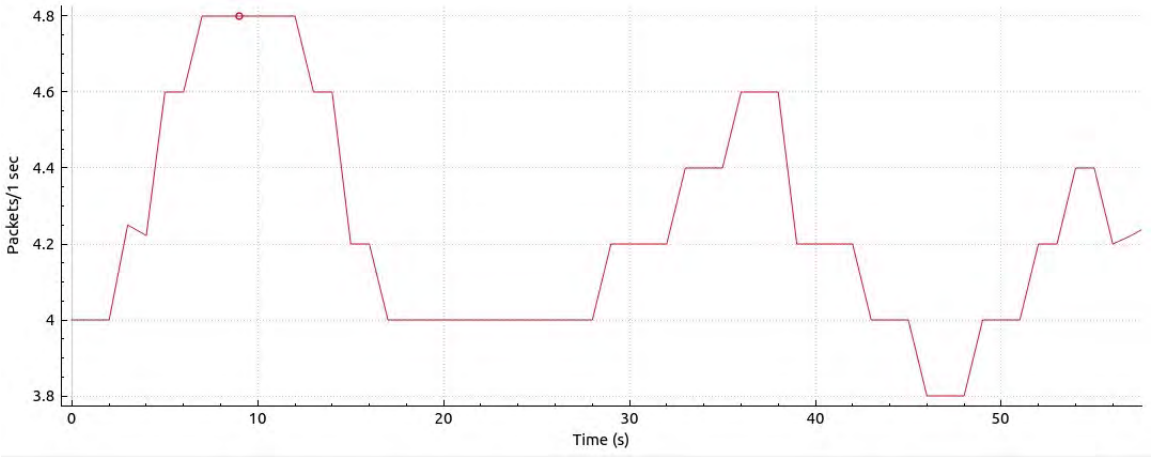


Figure 3.16: Packet Rate in Normal/Benign Traffic

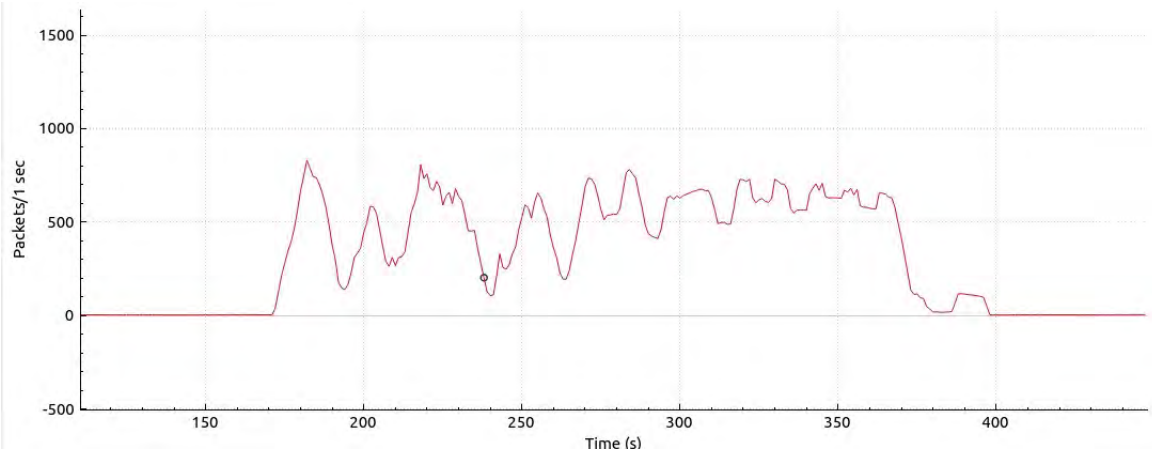


Figure 3.17: Packet Rate during ICMP Flood Attack

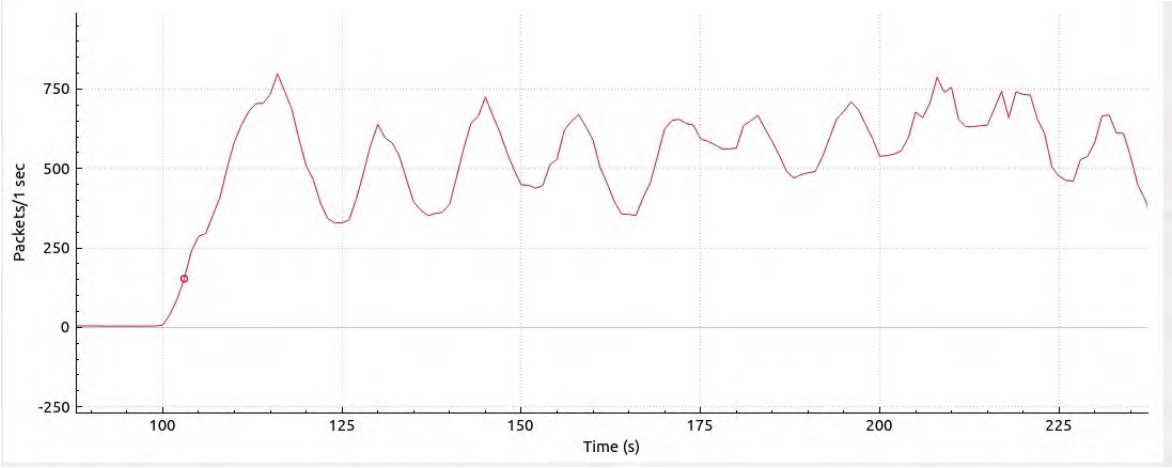


Figure 3.18: Packet Rate during UDP Flood Attack

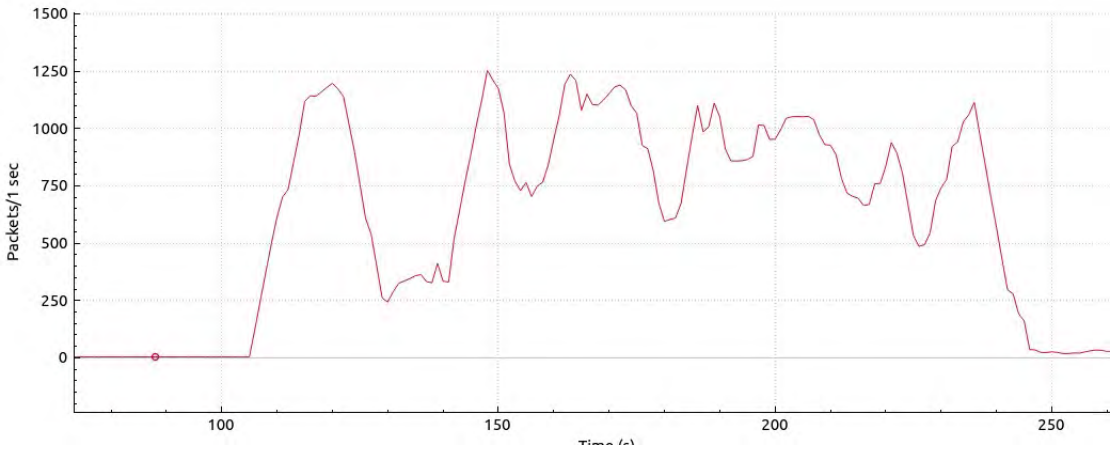


Figure 3.19: Packet Rate during TCP-SYN Flood Attack

Chapter 4

Methodology

4.1 Workflow

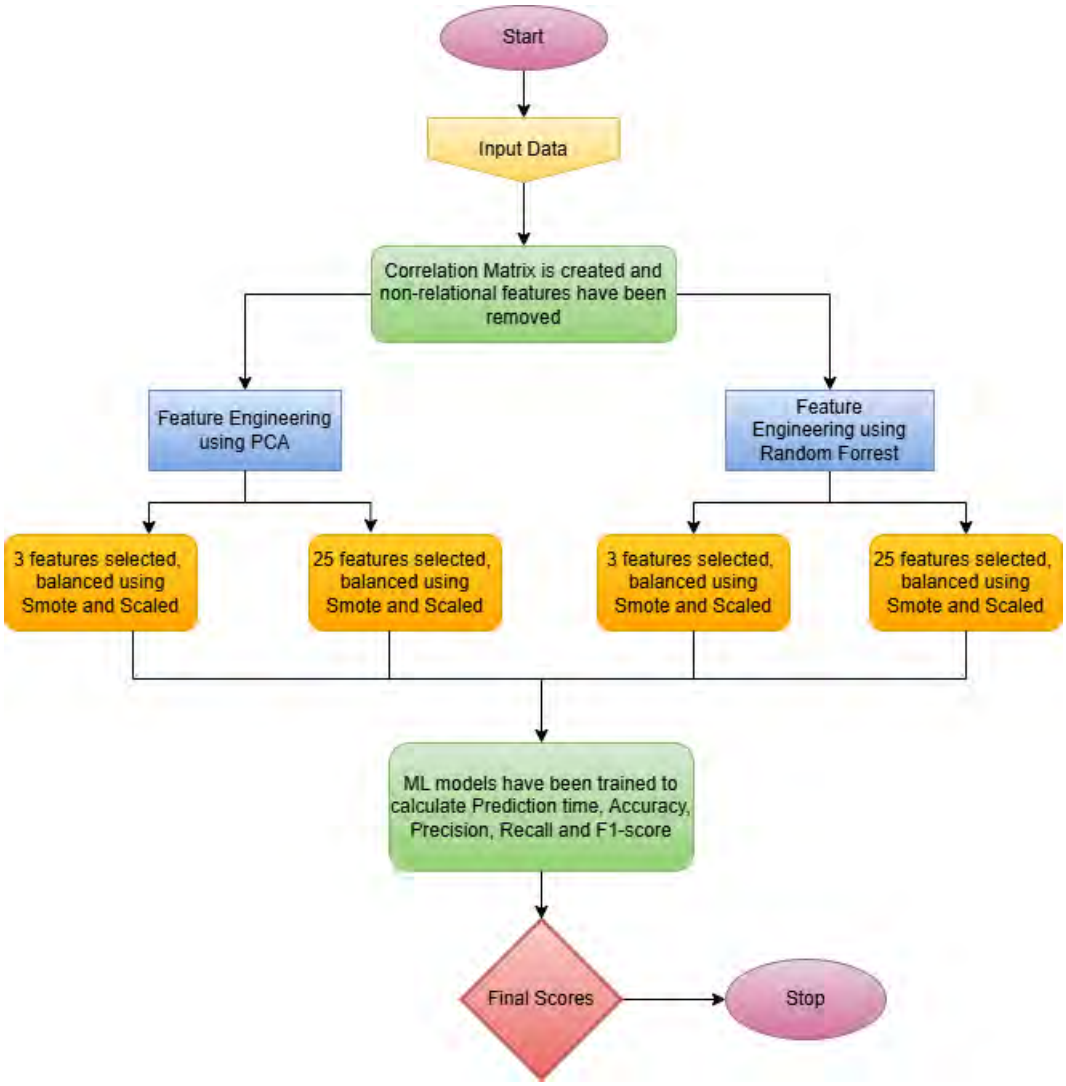


Figure 4.1: Workflow Diagram With Traditional Dataset

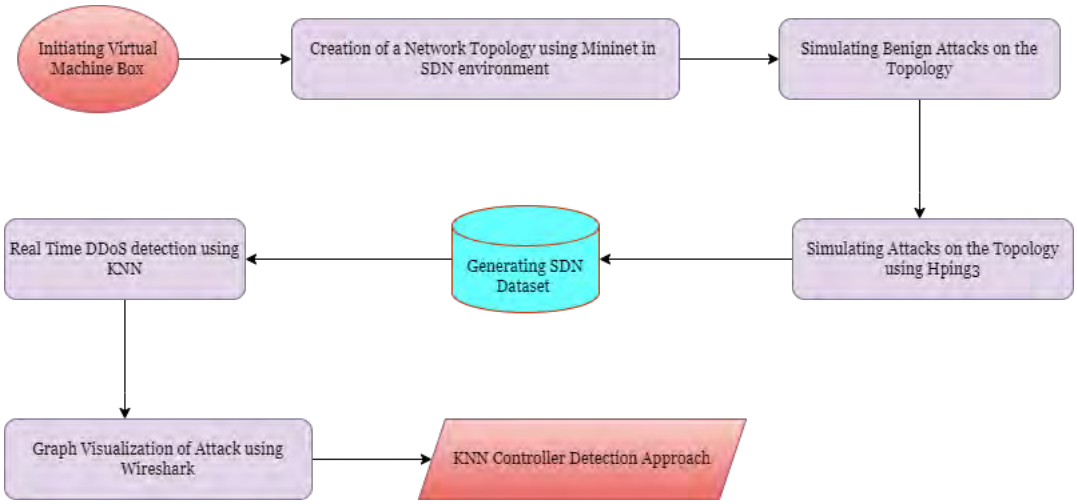


Figure 4.2: Workflow Diagram of SDN Environment

4.2 Model Analysis

4.2.1 Random Forest

Random forest is a supervised machine learning model that has a subtle feature selection that uses embedded techniques to work with. Feature selection is facilitated by a determination of how many features can be tested at each node in the tree [12]. This machine learning model can be used for both supervised and unsupervised learning techniques. The basis for it is the idea of ensemble learning, which is the process of merging several classifiers to solve a complex problem while improving the model's functionality. The higher the number of trees in the forest, the higher the accuracy which can solve the overfitting issue. As our research purpose is mostly based on classification problems, RF can be a good fit for implementation so we adopted the research using a random forest algorithm. To improve predicting accuracy and avoid overfitting, it includes generating several decision trees during training and integrating their results. The classification formula for RF can be expressed as,

$$RF(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Where $RF(x)$ is the final outcome for random forest prediction for x instances. N represents the number of decision tree. $Ti(x)$ represents the prediction of the ith tree in the random forest for x input. Figure 4.3 shows the basic Random Forest model architecture with N number of decision trees.

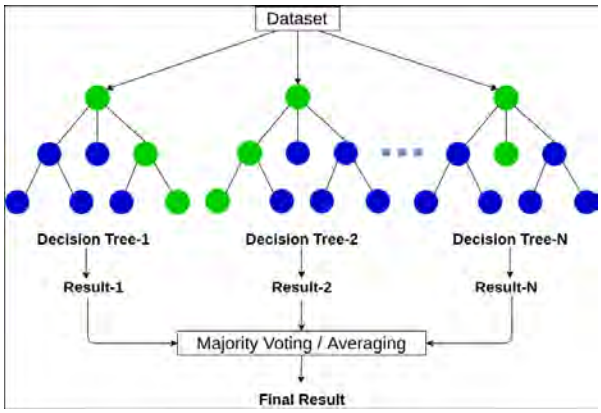


Figure 4.3: Random forest architechture

4.2.2 KNN (K-Nearest Neighbor)

KNN is an algorithm utilized for classification and regression tasks in supervised machine learning. The fundamental concept of KNN is predicting the class value of a data point by examining its 'k' nearest neighbors in the feature space. The KNN algorithm just memorizes the whole training dataset during the training process to store the feature vectors. KNN determines the most frequent class label among these 'k' neighbors during classification. While in regression, KNN calculates the mean of the target values. The choice of 'k' is crucial in prediction analysis. If 'k' is too small, it could cause predictions to be noisy, and if it's too big, it could make the decision limit too smooth. As our motive is mostly based on classification, KNN can be applicable as it is simple to implement, inherently multi-classification and does not make any assumptions about labeled data. Moreover, our dataset includes instances with features and class labels. The classification formula for KNN algorithm can be expressed as follows:

$$d\left(p,q\right)=\sqrt{\sum i=1^{n}\left(qi-p_{i}\right)^{2}}$$

- n is the dimension of the space
- pi is the coordinates of p in ith dimension
- qi is the coordinates of q in ith dimension

4.2.3 Decision Tree (DT)

Among all the supervised machine learning models, the Decision Tree algorithm has put its place in one of the most used and popular ones. It is a basic machine learning technique used for both regression and classification applications. It works by recursively dividing a dataset into smaller and smaller subsets according to specific feature values. The result is a tree-like model that depicts choices and potential outcomes. Similar to how branches grow from a tree's stem, every decision made in this process opens up a new set of possibilities, simulating human decision-making. A decision tree may be built really quickly in comparison to other classification techniques. SQL statements derived from trees can be quickly and readily used to access databases.

Comparing decision tree classifiers to other classification techniques, they achieve comparable, and occasionally even higher, accuracy. Depending on the amount of data, the amount of memory space on the computer, and the algorithm's scalability, decision tree algorithms can be implemented serially or in parallel [43]. Figure 4.4 shows the decision tree classifier algorithm with root node and leaf nodes.

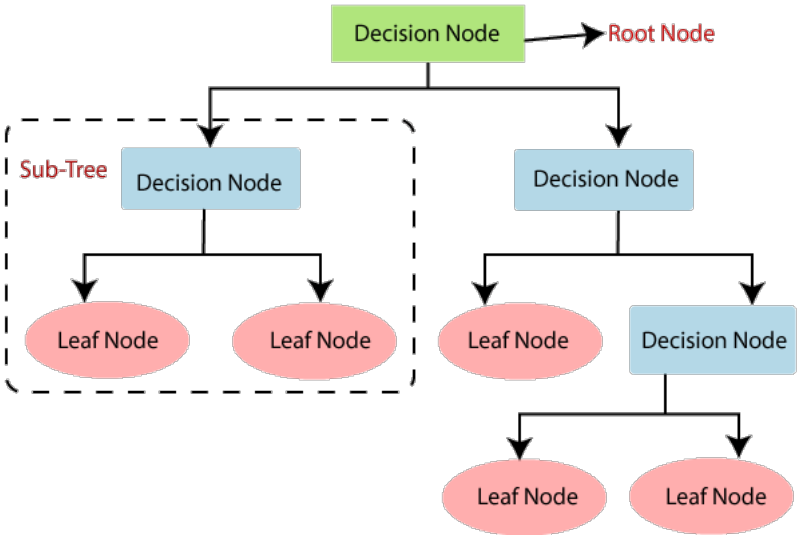


Figure 4.4: Decision Tree Classification Algorithm

Root Node: The decision tree originates at the root node. It depicts the complete dataset, which is then split up into two or more sets of similar data.
Leaf Node: After obtaining a leaf node, the tree cannot be further divided; leaf nodes are the ultimate output nodes.
Splitting: The process of splitting the decision node/root node into sub-nodes in accordance with the specified parameters is known as splitting.
Branch/Sub Tree: A tree created by slicing another tree into a branch or subtree.
Pruning: Removing undesirable limbs from a tree is the process of pruning.
Parent/Child Node: Nodes in a tree are referred to as parent and child nodes, respectively. The parent node is the root node.
When a dataset element is randomly labeled, the gini impurity calculates the frequency that it will be mislabeled. The formula for gini impurity has been shown below:

$$Gini(t) = 1 - \sum_{i=1}^{n} (p_i)^2$$

Here,

- p_i is the proportion of the ith class label
- t is the node
- n is the number of classes the node t contains

Entropy is a measure of information that expresses the disorder of the target's features. The feature with the least amount of entropy determines the ideal split, just like the Gini Index does. When the probability of the two classes is equal, it reaches its greatest value, and a node is considered pure when the entropy is at its lowest, or zero. The formula for entropy is given below:

$$Entropy(t) = -\sum_{i=1}^{n} p_i \, log_2 \, (p_i)$$

Here,

- t is the node
- N is the number of classes of the node t
- P_i is the proportion of instances of class i

4.2.4 Logistic Regression

Logistic regression is a statistical method that is widely used in machine learning for binary classification applications. It uses the values of one or more predictor factors to model the likelihood of the outcome variable such as whether an email is spam or not [29]. Simple yet effective, logistic regression is a powerful technique with a high degree of accuracy across a broad range of applications. It is frequently used as a foundation model for more intricate machine learning models. To ascertain situations when there are more than two possible outcomes, logistic regression can also be utilized. In this kind of situation it is known as multinomial logistic regression. Applications of logistic regression that are well-known include cancer diagnosis, fraud detection, and email spam filtering [40]. As logistic regression is computationally efficient, it makes it easier and more feasible to implement on real time detection such as the dataset we have generated using real time data. Predictions and their probability are mapped using a logistic function known as the sigmoid function. It is an S-shaped curve which transforms any real number into a range between 0 and 1. [46] Moreover, the model predicts that the instance belongs to that class if the estimated probability produced by the sigmoid function exceeds a predetermined threshold on the graph. The model anticipates that the instance does not belong in the class if the calculated probability is less than the predetermined threshold.

The following equation represents the sigmoid function:

$$f \ (x) \ = \ \frac{1}{1 + e^{-x}}$$

Here,

- e is the natural logarithm base
- x is the input value

The equation represents logistic regression:

$$f \ (x) \ = \ \frac{e^{(b_0 + b_1 X)}}{1 + e^{(b_0 + b_1 X)}}$$

Here,

- x is the input value
- y is the predicted output
- b0 indicates the bias or intercept term
- b1 represents the coefficient for input (x)

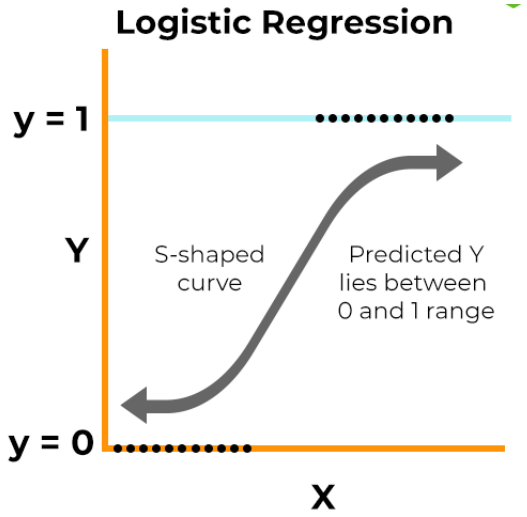


Figure 4.5: S-shaped Curve for Logistic Regression

Figure 4.5 shows an S-shaped curve for the LR classifier where the predicted Y lies between the ranges 0 and 1. By identifying patterns and relationships in the network traffic data, logistic regression assists in the detection of DDoS attacks by offering a probabilistic framework for categorizing newly discovered, unseen data as either regular traffic or potentially malicious DDoS activity.

4.2.5 Naive Bayes

Naive Bayes is a probabilistic classifier that relies on the premise of feature independence and is based on Bayes' Theorem. By categorizing network traffic data as either normal or an attack, it can be utilized efficiently to detect DDoS (Distributed Denial of Service) attacks. The class with the highest probability is chosen by naive Bayes classifiers, which determine the probability of each data point belonging to each class. Given the class variable, it is assumed that the existence (or lack) of a given feature is independent of the presence (or absence) of any other feature. This approach has the benefit of having a simple algorithm with little computational complexity. On the other hand, the weakness of using it is that it will affect the accuracy because it is not always possible to apply the Naive Bayes feature independently [14].
The formula for Naive Bayes is given below:

$$P(C \mid x) = \frac{P(x \mid C) \cdot P(C)}{P(x)}$$

Here,

- $P(C \mid x)$ is the posterior probability of the hypothesis given that the evidence is true
- $P(x \mid C)$ is the likelihood of evidence given that the hypothesis is true
- $P(C)$ is the prior probability
- $P(x)$ is the prior probability that the evidence is true

In our research, both benign and DDoS attacks can be classified by the naive Bayes algorithm. It has been demonstrated that by achieving a higher degree of detection accuracy than the model without the tuning process, the hyper parameter tuning procedure contributes to the improvement of the predictive model.

4.3 Real Time Network Simulation and DDoS Attack Detection

In the SDN controller.py script, we used the KNN classifier model for detection. At first, we trained the generated dataset in the controller. After that when we ran the topology there was a CSV file generated in the directory. When we ran the KNN algorithm in the controller it gave us 100% accuracy in the flow training.

In this file, when we were pinging the hosts, all the data that we were fetching was generated there. For checking normal traffic and DDoS traffic, we considered host1 as the server. Furthermore, we opened three random hosts. Among them, for two hosts, we ping normal traffic to the host. In Figure 4.6 we simulate benign traffic.

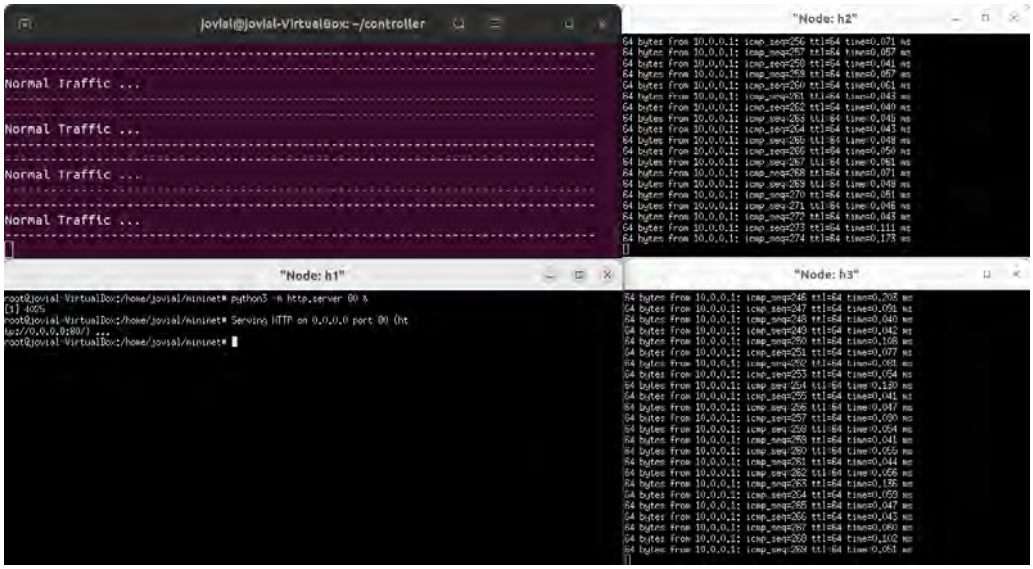


Figure 4.6: Normal/Benign Traffic Simulation

When we were using the Hping3 tool the controller detected the DDoS and also detected the host which was affected. In Figure 4.7, 4.8 and 4.9 we see different

DDoS attacks detection using KNN controller.

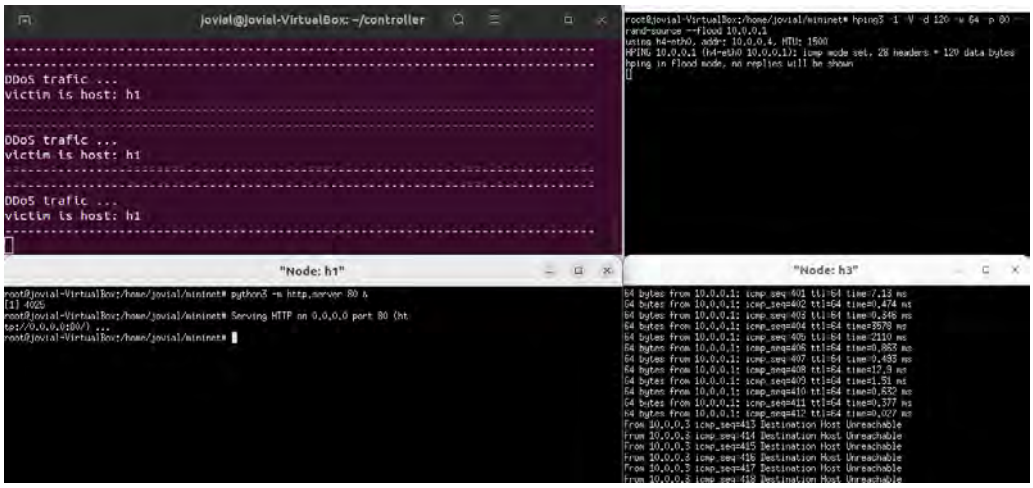


Figure 4.7: ICMP Flood Attack Detection using KNN

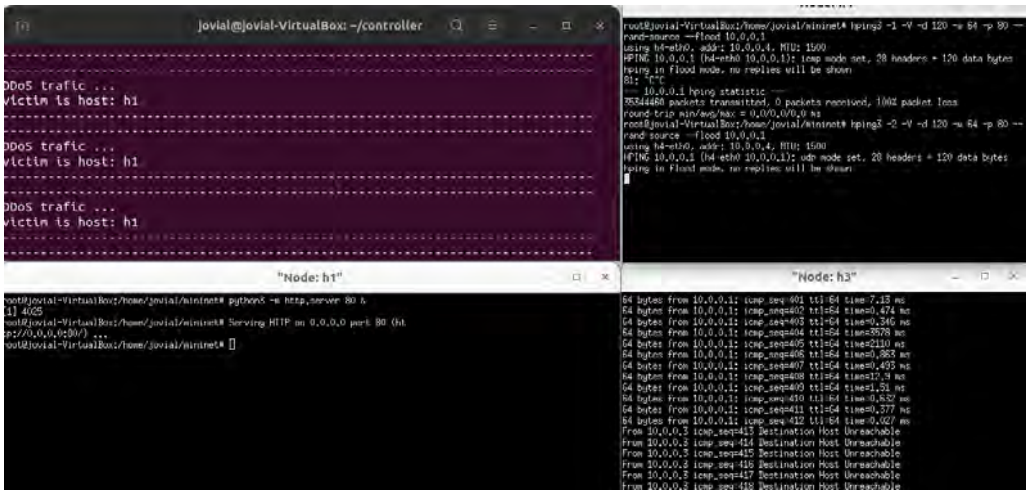


Figure 4.8: UDP Flood Attack Detection using KNN

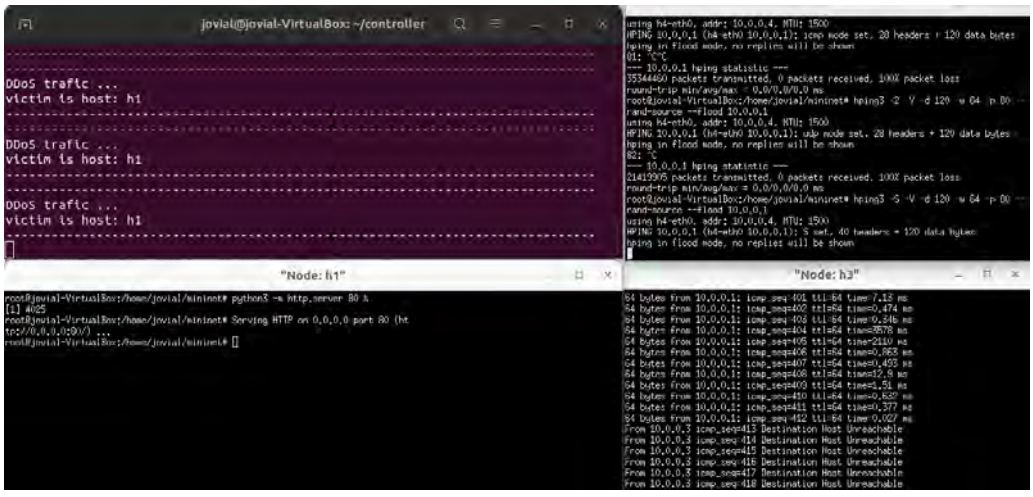


Figure 4.9: TCP-SYN Flood Attack Detection using KNN

Chapter 5

Result Analysis

The performance of different machine learning approaches is based on 4 different metrics known as accuracy, precision, recall, prediction time and F1 score. The evaluation of performance is based on the values of True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). In our analysis, we use all these measurements to analyze the performance of different machine and deep learning models.

Accuracy: The accuracy explains the total number of correctly predicted values against the total prediction value explored. Its a blunt measure that can sometimes be misleading. The formula for accuracy is:

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{1}$$

Precision: It explains total correctly evaluated value against the number of positive predicted values The formula for precision is:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall: It explains the ratio of total correctly predicted instances against total positive instances. The formula for recall is:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F1-Score: It's a measure of combining recall and precision. There is a trade-off between precision and recall. F1-Score can measure the effectiveness of this trade-off. The equation of F1-Score is:

$$F1 - score = \frac{2 * recall * precision}{recall + precision} \tag{4}$$

Training Time: The duration required for the model to learn from the training dataset. It is measured in seconds or minutes and indicates the computational efficiency of the training process.

Model Size: The storage space occupied by the trained model, usually measured in kilobytes (KB) or megabytes (MB). Smaller model sizes are preferable for easier deployment and faster loading times.

Memory Usage: The amount of RAM used by the model during training and inference, measured in megabytes (MB). Efficient memory usage is crucial for handling large datasets and ensuring the model can run on hardware with limited resources.

Prediction Time: The time taken by the model to make predictions on new data, typically measured in seconds. Faster prediction times are essential for real-time applications.

Confusion Matrix: A table that displays the performance of a classification model by showing the counts of:

True Positives (TP): Correctly predicted positive cases.
True Negatives (TN): Correctly predicted negative cases.
False Positives (FP): Incorrectly predicted positive cases.
False Negatives (FN): Incorrectly predicted negative cases.

5.1 Model Performance Analysis

In this study, we compare various machine learning models for DDoS attack detection using efficient features selected through Principal Component Analysis (PCA) and Random Forest (RF). We evaluated these models on several performance metrics, including Accuracy, Precision, Recall, F1 score, Training Time, Model Size, Memory Usage, and Prediction Time. We tested with both 3-feature and 25-feature sets selected by PCA and RF to understand the impact of feature selection on model performance.

5.1.1 25 Features Selected by PCA Analysis

In Table 5.1 when using PCA for selecting 25 features, the Decision Tree model outperformed others with an accuracy of 0.9963 and it also maintained a moderate training time of 21.6533 seconds. KNN achieved slightly higher accuracy which is 0.9941 compared to using 3 features, but its prediction time increased significantly 200.0509 seconds. The Random Forest (RF) model showed considerable improvement in accuracy 0.9872 with a manageable training time 57.4725 seconds, although it had an increased prediction time 0.6571 seconds.

Metric	RF	KNN	LR	DT	NB
Accuracy	0.9872	0.9941	0.9363	0.9963	0.8575
Precision	0.9875	0.9941	0.9414	0.9963	0.8947
Recall	0.9872	0.9941	0.9363	0.9963	0.8585
F1 Score	0.9873	0.9941	0.9360	0.9963	0.8495
Training Time	57.4725 s	1.3080 s	258.0553 s	21.6533 s	0.2961 s
Model Size	3546.11 KB	78.09 MB	4.12 KB	383.89 KB	6.66 KB
Memory Usage	2767.04 MB	2817.55 MB	2675.16 MB	2610.59 MB	2612.95 MB
Prediction Time	0.6571 s	200.0509 s	0.666 s	0.0737 s	0.0787 s

Table 5.1: 25 feature selection based on PCA's model performance

5.1.2 3 Features Selected by PCA Analysis

But on the other hand, in Table 5.2 when using PCA for selecting 3 features, the KNN model achieved the highest accuracy 0.9808 and F1 score 0.9808 with a relatively low training time of 0.0994 seconds but a high prediction time 7.1450 seconds. The Decision Tree also performed well with an accuracy of 0.9772 and a minimal prediction time 0.0275 seconds. On the other hand, the Logistic Regression model showed the longest training time 246.9014 seconds and had the lowest accuracy 0.8628 among the models.

Metric	RF	KNN	LR	DT	NB
Accuracy	0.9471	0.9808	0.8628	0.9772	0.7881
Precision	0.9487	0.9810	0.8639	0.9773	0.8138
Recall	0.9471	0.9808	0.8628	0.9773	0.7890
F1 Score	0.9475	0.9808	0.8604	0.9773	0.7741
Training Time	18.96s	0.0994s	246.90s	7.41s	0.1707s
Model Size	2199.14KB	17.40MB	1.66KB	2569.92KB	1.95KB
Memory Usage	2363.36MB	2566.33MB	2025.15MB	1993.61MB	2012.31MB
Prediction Time	0.3310s	7.1450s	0.0399s	0.0275s	0.1329s

Table 5.2: 3 feature selection based on PCA's model performance

5.1.3 25 Features Selected by RF Analysis

Moreover, In case of Random Forest for while inspecting Table 5.3 selecting 25 features, the Decision Tree model again achieved the highest accuracy of 0.9963. The KNN model showed high accuracy 0.9941 but suffered from a very high prediction time 200.0509 seconds. The Random Forest model maintained high accuracy 0.987 but had increased training times of 57.4725 seconds and prediction times of 0.6571 seconds.

Metric	RF	KNN	LR	DT	NB
Accuracy	0.9872	0.9941	0.9363	0.9963	0.8575
Precision	0.9875	0.9941	0.9414	0.9963	0.8947
Recall	0.9872	0.9941	0.9363	0.9963	0.8585
F1 Score	0.9873	0.9941	0.9360	0.9963	0.8495
Training Time	57.4725 s	0.4347 s	258.0553 s	21.6533 s	0.2961 s
Model Size	2666.75 KB	78.09 MB	4.12 KB	383.89 KB	6.66 KB
Memory Usage	2363.36 MB	2566.33 MB	2675.16 MB	2610.59 MB	2612.95 MB
Prediction Time	0.6571 s	200.0509 s	0.0444 s	0.0275 s	0.1329 s

Table 5.3: 25 feature selection based on RF's model performance

5.1.4 3 Features Selected by RF Analysis

For the 3 features selected in Table 5.4 using RF, the Decision Tree model achieved near-perfect performance with an accuracy of 0.9997 and very low training 1.3853 seconds and prediction times 0.0167 seconds. The Random Forest model also performed exceptionally well with an accuracy of 0.9980 and a training time of 17.4986 seconds. However, the Logistic Regression (LR) and Naive Bayes (NB) models showed poor performance, with accuracies of 0.4040 and 0.2538, respectively.

Metric	RF	KNN	LR	DT	NB
Accuracy	0.9980	0.9935	0.4040	0.9997	0.2538
Precision	0.9980	0.9936	0.3786	0.9997	0.2741
Recall	0.9980	0.9935	0.4040	0.9997	0.2533
F1 Score	0.9980	0.9935	0.3529	0.9997	0.2173
Training Time	17.4986 s	0.0994 s	122.3028 s	1.3853 s	0.1462 s
Model Size	2199.14 KB	17.40 MB	1.67 KB	79.31 KB	1.97 KB
Memory Usage	2125.25 MB	2149.16 MB	2149.06 MB	2149.06 MB	2149.06 MB
Prediction Time	0.2889 s	8.1489 s	0.0163 s	0.0167 s	0.0383 s

Table 5.4: 3 feature selection based on RF's model performance

The confusion matrices in Figures 5.1, 5.2 and 5.3 provides a detailed breakdown of our primary 3 model's performance using 3 important features and by showing the counts of true positives, true negatives, false positives, and false negatives. This allows us a deeper understanding on identifying specific areas of strengths and weaknesses.

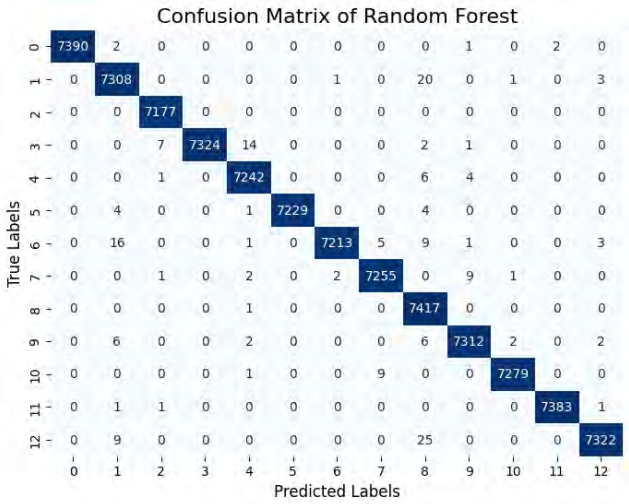


Figure 5.1: Confusion Matrix of RF with 3 Features

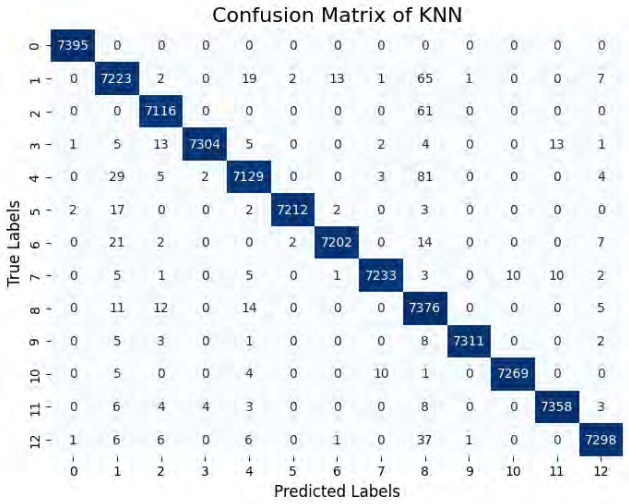


Figure 5.2: Confusion Matrix of KNN with 3 Features

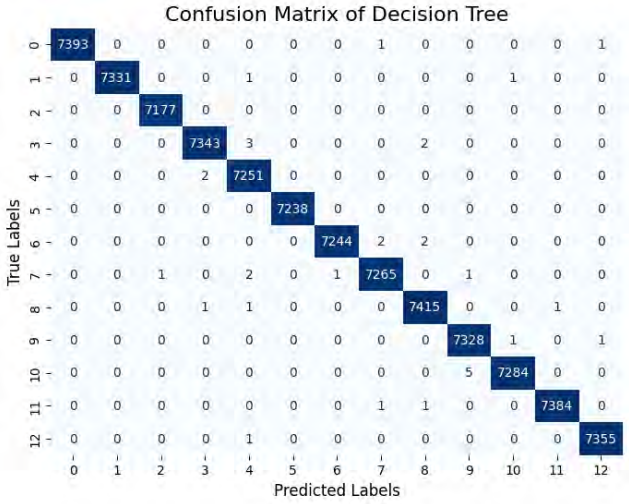


Figure 5.3: Confusion Matrix of Decision Tree with 3 Features

We further analyzed some learning curves for some machine learning models DT, KNN and RF models to figure out the overfitting issues of our study. If the training score and cross-validation score match, we can conclude that we solved the model's overfitting issue eventually.

Figure 5.4 shows a learning curve for a Decision Tree Classifier. The training score stays at a perfect 1.0 which indicates the model fits the training data perfectly. The cross-validation score starts at about 0.9 and slowly gets better as more training examples are added, eventually matching the training score. This gap between the training and cross-validation scores indicates that the model starts off overfitting but improves as it gets more data.

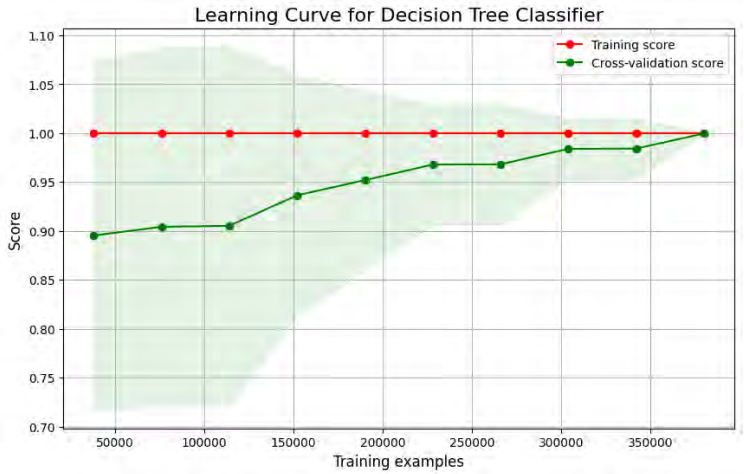


Figure 5.4: Learning Curve for Decision Tree Classifier

The learning curve in Figure 5.5 for the KNN classifier shows that the training score remains consistently high near 1.0 which indicates the model is perfectly fitting the training data. The cross-validation score improves gradually with more training examples which starting from around 0.85 and approaching the training score as the dataset size increases.

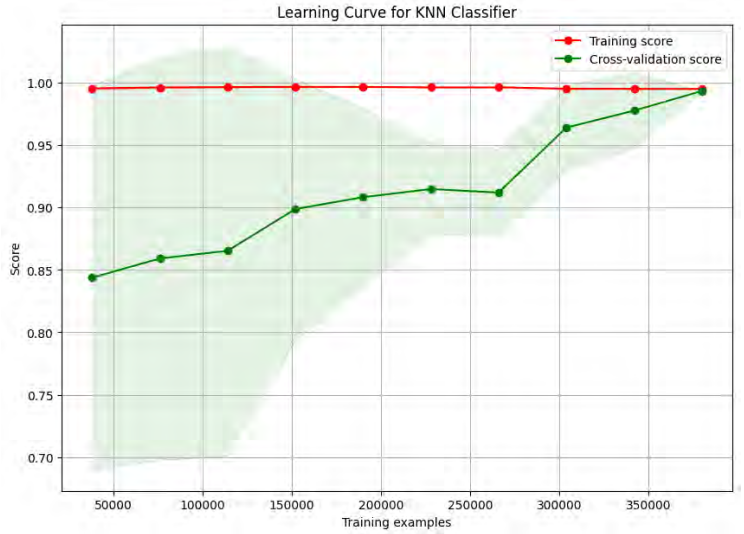


Figure 5.5: Learning Curve for KNN Classifier

In Figure 5.6 the Random Forest classifier shows a training score that is consistently higher near 1.0, indicating the model is fitting the training data very well. The cross-validation score improves steadily with the increase in training examples, starting around 0.90 and gradually converging towards the training score.

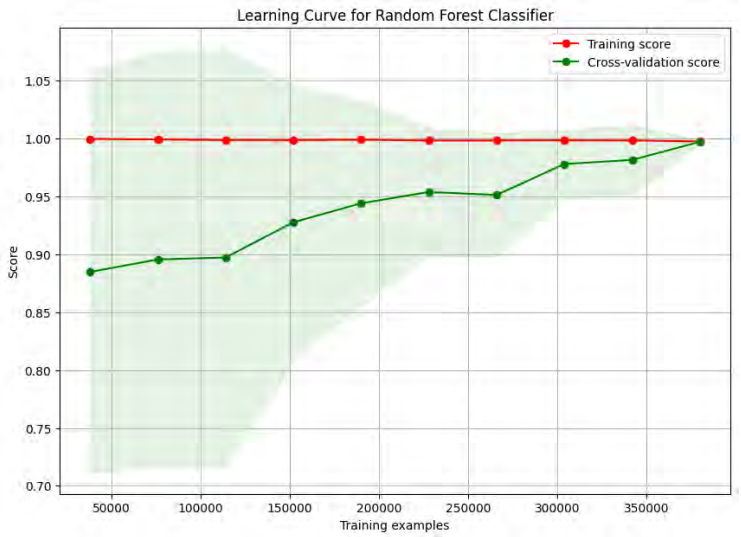


Figure 5.6: Learning Curve for Random Forest Classifier

When comparing 3 feature selection to 25 feature selection, models with just 3 features performed well in terms of training time, model size, memory usage, and prediction time. At the same time, some models possessed high accuracy. The Decision Tree and Random Forest models had almost perfect accuracy and much faster training and prediction times when feature selection was based on RF feature importance. Again, a few models also performed well when feature selection was based on PCA analysis. This shows that using fewer, well-chosen features can make models quicker and more efficient without losing accuracy. Fewer features are better for practical use because they save time and resources. Therefore, we found that selecting fewer features based on Random Forest feature importance is more significant than PCA, as it doesn't damage model accuracy and perhaps improves accuracy in the cases of Random Forest, KNN, and Decision Tree models, making those models more optimized and resource-efficient as well. Lastly, based on the Random Forest importance selection of three features, the Decision Tree model provided the best overall performance among all the models.

5.2 Real Time DDoS Detection using KNN Controller in SDN Network

During DDoS attack in our SDN environment, we prepare a KNN controller model that can successfully detect the DDoS attack with great accuracy which is 100 percent. It perhaps adds great dynamism to our study as it can detect the attack in real-time. The confusion matrix, as depicted in Figure 5.7, confirms a perfect success rate with an accuracy of 100% indicating that the model correctly identified benign traffic and DDoS attack instances without any false positives or negatives.

The training process was efficient, and completed in just 24.5 seconds, which underscores the model's suitability for real-time applications. The success accuracy was 100.00%, and the failure accuracy was 0.00% demonstrating the model's robustness and reliability. This approach is an excellent choice for real-time DDoS detection in SDN networks.

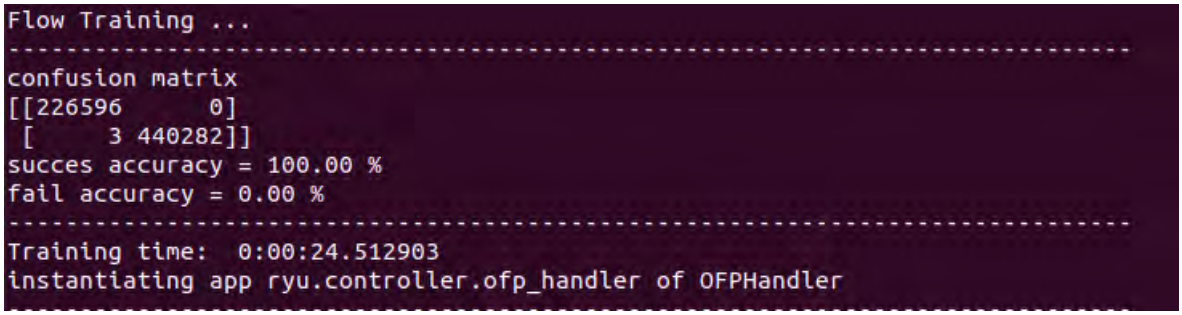


Figure 5.7: Real-time DDoS detection using KNN

5.3 Comparative Analysis

Research Study	**No. Of Selected Features (out of 47)**	**Feature Selection Method**	**DT**	**RF**	**KNN**	**LR**	**NB**	**SDN Env.**
Jony et al., 2024 [39]	34	✗	✗	✗	✗	✗	✗	✗
Raju et al., 2023 [33]	20	CNN and LSTM	✓	✓	✗	✓	✓	✗
Barletta et al., 2023 [24]	13	PCA	✗	✗	✗	✗	✗	✗
Pirtama et al., 2023 [31]	5	PCA	✗	✓	✗	✗	✗	✗
Proposed Approach	**3**	**PCA and Random Forest**	✓	✓	✓	✓	✓	✓

Table 5.5: Comparison With Other Researches Regarding CICIoT2023 Dataset

In the Table 5.5, we depicted a comparison on feature extraction for the CICIoT2023 dataset with some other existing papers. Other papers extracted respective no. of features and trained those features by machine learning algorithms on the basis of

only single feature selection method. On the contrary, we derived 3 important features out of 47 features of the mentioned dataset on the basis of 2 selection methods - PCA and Random Forest. Moreover, among the 4 existing papers for CICIoT2023 dataset, only one paper has successfully applied more than 3 or 4 machine learning techniques; but the paper's authors extracted about 20 features out of 47 features to obtain decent accuracies.

Our paper is distinctive as we have achieved the highest accuracy across all the models (Decision Tree, Random Forest, Logistic Regression, Naive Bayes and KNN) by only extracting 3 features. Most of the works for this dataset is solely based on one single feature selection method. But Random Forest has the best performance for feature selection and only our paper implements Random Forest algorithm to derive most efficient approach to feature engineering.

5.4 Limitations of Our Study

Our research faced some challenges due to the nature of our dataset. For instances,

- Our SDN dataset is limited because it was generated in a controlled setting and on a smaller scale, which might not represent the full range of complicated attack scenarios found in real-world SDN environments.
- In comparison to larger and more diverse datasets like CICIoT2023, our SDN dataset might fall short in terms of variety and scope, potentially affecting our findings.
- The synthetic nature of our SDN dataset may not fully reflect the intricacies and unpredictability of real network traffic.
- Despite extensive feature engineering, few machine learning models showed lower-than-expected performance metrics in terms of accuracy, recall, precision, and F1-score.
- Our SDN dataset was generated within limited courses, which may hinder the adequacy of machine learning models because models typically require large and diverse datasets.

Chapter 6

Future Work

As a future improvement, this study can be used for DDoS attack mitigation strategies. There are several findings from our analysis of different mitigation methods in the case of real-time DDoS detection. For example:

- The mitigation involves using the OpenFlow protocol to block malicious packets. OpenFlow allows network administrators to control the flow of traffic by defining rules for packet handling. By setting specific rules, administrators can identify and block packets that match known malicious patterns or behaviors, effectively preventing these packets from reaching their intended targets and mitigating potential security threats.

- This study can enhance network security by installing firewall applications on the SDN (Software-Defined Networking) controller. These applications can monitor and manage the traffic flowing through the network. If they detect any suspicious or harmful traffic, they can block it, preventing it from causing damage or reaching its target. This approach uses the centralized control of SDN to make the network more secure and responsive to threats.

- This study can improve network security by setting up packet filtering, which checks the source and destination of each data packet. If a packet comes from or is going to a suspicious or unauthorized location, it gets blocked. This helps prevent harmful traffic from entering or leaving the network, keeping it safe from attacks.

- This study can enhance network security by creating a blacklist of specific IP addresses or types of traffic known to be harmful. Any traffic matching this blacklist is automatically blocked from entering or moving within the network. This helps in quickly identifying and stopping malicious activities.

Our research can be improved much more by implementing some combined/hybrid mitigation strategies that can be incorporated into real-time scenarios.

Chapter 7

Conclusion

The sharp rise of DDoS attacks are still getting attention in the software-defined networks. Due to the inefficient maintenance of these networks, cybercriminals are able to easily infiltrate these networks and launch various kinds of cyber attacks where DDoS plays the most vital role. Thus, the primary goal of our research is to identify this threat spontaneously. By performing an in-depth analysis of methodologies, we have manifested the significance of Machine Learning algorithms in detecting DDoS. Two types of datasets have been used in the analysis:
At first, we performed data pre-processing in the CICIoT2023 dataset using Random Forest and the Principal Component Analysis (PCA) classifiers. Feature engineering has been done using both classifiers. Secondly, machine learning models including Random Forest, KNN, Decision Tree, Logistic Regression and Naive Bayes have been implemented in order to detect and categorize the DDoS attacks. We have received significantly promising results including accuracy, precision, recall prediction time and F1-score. A high accuracy has been achieved after the use of these models.
For the second dataset, we have produced a network topology having 18 hosts and 6 switches using a controller named Mininet in the SDN environment. Attacks have been simulated on the topology from the host. Using the topology, the SDN dataset consisting of real time attacks have been formed where the attacks are detected by using KNN, returning high accuracy, instantly. Then, by the use of WireShark, flow graphs are generated which represent the flow of packets per second during the attack. Thus, our study looks forward to simulating a real-time DDoS attack in a software-defined network and accurately detecting malicious and benign traffic using KNN architecture. Additionally, it evaluates several machine learning models while using traditional data on IoT devices with greater efficiency and accuracy to detect DDoS attacks. Moreover, the real time DDoS attack detection is also going to benefit us in initiating the mitigation of cyber attacks in future.

Bibliography

[1] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014. DOI: 10.1109/COMST.2014.2326417.

[2] K. Kaur, J. Singh, and N. Ghumman, "Mininet as software defined networking testing platform," Aug. 2014.

[3] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015. DOI: 10.1109/JPROC.2014.2371999.

[4] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," en, *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, Feb. 2017.

[5] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 325–346, 2017. DOI: 10.1109/COMST.2016.2618874.

[6] L. Wang and J. Yang, "A research survey in stepping-stone intrusion detection," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, Dec. 2018. DOI: 10.1186/s13638-018-1303-2.

[7] E. T. f. t. arXiv, *The first ddos attack was 20 years ago. this is what we've learned since.* Apr. 2020. [Online]. Available: https://www.technologyreview.com/2019/04/18/103186/the-first-ddos-attack-was-20-years-ago-this-is-what-weve-learned-since/.

[8] Y. Gautam, B. P. Gautam, and K. Sato, "Experimental security analysis of sdn network by using packet sniffing and spoofing technique on pox and ryu controller," in *2020 International Conference on Networking and Network Applications (NaNA)*, 2020, pp. 394–399. DOI: 10.1109/NaNA51271.2020.00073.

[9] K. Shaukat Dar, S. Luo, S. Chen, and D. Liu, "Cyber threat detection using machine learning techniques: A performance evaluation perspective," Oct. 2020. DOI: 10.1109/ICCWS48432.2020.9292388.

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

[11] K. Doshi, Y. Yilmaz, and S. Uludag, "Timely detection and mitigation of stealthy ddos attacks via iot networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2164–2176, 2021. DOI: 10.1109/TDSC.2021.3049942.

[12] V. Gaur and R. Gujral, "Analysis of machine learning classifiers for early detection of ddos attacks on iot devices," *Arabian Journal for Science and Engineering*, vol. 47, Jul. 2021. DOI: 10.1007/s13369-021-05947-3.

[13] H. Kousar, M. M. Mulla, P. Shettar, and D. G. Narayan, "Detection of ddos attacks in software defined network using decision tree," in *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, 2021, pp. 783–788. DOI: 10.1109/CSNT51715.2021.9509634.

[14] F. F. Setiadi, M. W. A. Kesiman, and K. Y. E. Aryanto, "Detection of dos attacks using naive bayes method based on internet of things (iot)," *J. Phys. Conf. Ser.*, vol. 1810, no. 1, p. 012 013, Mar. 2021.

[15] M. Tawfik, N. M. Al-Zidi, B. Alsellami, A. M. Al-Hejri, and S. Nimbhore, "Internet of things-based middleware against cyber-attacks on smart homes using software-defined networking and deep learning," in *2021 2nd International Conference on Computational Methods in Science Technology (ICCMST)*, 2021, pp. 7–13. DOI: 10.1109/ICCMST54943.2021.00014.

[16] A. Albu-Salih, "Performance evaluation of ryu controller in software defined networks," *Journal of Al-Qadisiyah for Computer Science and Mathematics*, vol. 14, p. 1, Feb. 2022. DOI: 10.29304/jqcm.2022.14.1.879.

[17] M. F. Ashfaq, M. Malik, U. Fatima, and M. K. Shahzad, "Classification of iot based ddos attack using machine learning techniques," in *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2022, pp. 1–6. DOI: 10.1109/IMCOM53663.2022.9721740.

[18] M. Aslam, D. Ye, A. Tariq, *et al.*, "Adaptive machine learning based distributed Denial-of-Services attacks detection and mitigation system for SDN-enabled IoT," en, *Sensors (Basel)*, vol. 22, no. 7, p. 2697, Mar. 2022.

[19] R. K. Chouhan, M. Atulkar, and N. K. Nagwani, "A framework to detect DDoS attack in ryu controller based software defined networks using feature extraction and classification," en, *Appl. Intell.*, Jun. 2022.

[20] V. Gaur and R. Kumar, "Ddoslstm: Detection of distributed denial of service attacks on iot devices using lstm model," in *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, 2022, pp. 01–07. DOI: 10.1109/IC3IOT53935.2022.9767889.

[21] J. Li, T. Tu, Y. Li, S. Qin, Y. Shi, and Q. Wen, "DoSGuard: Mitigating denial-of-service attacks in software-defined networks," en, *Sensors (Basel)*, vol. 22, no. 3, p. 1061, Jan. 2022.

[22] A. Ahmim, F. Maazouzi, M. Ahmim, S. Namane, and I. B. Dhaou, "Distributed denial of service attack detection for the internet of things using hybrid deep learning model," *IEEE Access*, vol. 11, pp. 119 862–119 875, 2023. DOI: 10.1109/ACCESS.2023.3327620.

[23] A. A. Alahmadi, M. Aljabri, F. Alhaidari, *et al.*, "Ddos attack detection in iot-based networks using machine learning models: A survey and research directions," *Electronics*, vol. 12, no. 14, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12143103. [Online]. Available: https://www.mdpi.com/2079-9292/12/14/3103.

[24] V. S. Barletta, D. Caivano, M. De Vincentiis, A. Pal, and M. Scalera, "Hybrid quantum architecture for smart city security," 2023.

[25] M. A. GILL, N. AHMAD, M. KHAN, F. ASGHAR, and A. RASOOL, "Cyber attacks detection through machine learning in banking," *Bulletin of Business and Economics (BBE)*, vol. 12, no. 2, pp. 34–45, Aug. 2023. [Online]. Available: https://bbejournal.com/index.php/BBE/article/view/443.

[26] N. Gupta, S. Tanwar, and S. Behal, "Software defined network implements real time detection and mitigation of ddos attacks," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2023, pp. 1–7. DOI: 10.1109/ICCCNT56998.2023.10307802.

[27] I. Jemal, O. Cheikhrouhou, and M. A. Haddar, "Iot dos and ddos attacks detection using an effective convolutional neural network," in *2023 International Conference on Cyberworlds (CW)*, 2023, pp. 373–379. DOI: 10.1109/CW58918.2023.00065.

[28] Z. Liu, Y. Wang, F. Feng, Y. Liu, Z. Li, and Y. Shan, "A ddos detection method based on feature engineering and machine learning in software-defined networks," *Sensors*, vol. 23, no. 13, 2023, ISSN: 1424-8220. DOI: 10.3390/s23136176. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/6176.

[29] U. Maheswari, Vishnukumar, Meganathan, and S. C, "Detection and mitigation of DDoS attacks in network traffic using machine learning techniques," in *2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, Coimbatore, India: IEEE, Jun. 2023, pp. 1–6.

[30] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, 2023, ISSN: 1424-8220. DOI: 10.3390/s23135941. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/5941.

[31] A. Pirtama, Y. Prasetia, R. I. Saputra, and E. A. Winanto, "Improvement attack detection on internet of thinks using principal component analysis and random forest," *Media J. Gen. Comput. Sci.*, vol. 1, no. 1, pp. 14–19, Dec. 2023.

[32] V. S. A. Raju and S. B, "Network intrusion detection for iot-botnet attacks using ml algorithms," in *2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, 2023, pp. 1–6. DOI: 10.1109/CSITSS60515.2023.10334188.

[33] V. S. A. Raju and Suma, "Network intrusion detection for IoT-botnet attacks using ML algorithms," in *2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, Bangalore, India: IEEE, Nov. 2023, pp. 1–6.

[34] S. Sadhwani, B. Manibalan, R. Muthalagu, and P. Pawar, "A lightweight model for DDoS attack detection using machine learning techniques," en, *Appl. Sci. (Basel)*, vol. 13, no. 17, p. 9937, Sep. 2023.

[35] A. Sharma and H. Babbar, "Bot-iot: Detection of ddos attacks in internet of things for smart cities," in *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2023, pp. 438–443.

[36] A. K. Tahirou, K. Konate, and M. M. Soidridine, "Detection and mitigation of ddos attacks in sdn using machine learning (ml)," in *2023 International Conference on Digital Age Technological Advances for Sustainable Development (ICDATA)*, 2023, pp. 52–59. DOI: 10.1109/ICDATA58816.2023.00019.

[37] J. Wang, L. Wang, and R. Wang, "A method of ddos attack detection and mitigation for the comprehensive coordinated protection of sdn controllers," *Entropy*, vol. 25, no. 8, 2023, ISSN: 1099-4300. DOI: 10.3390/e25081210. [Online]. Available: https://www.mdpi.com/1099-4300/25/8/1210.

[38] M. Dandotiya and R. R. Singh Makwana, "Ddos attack detection and mitigation in sdn environment: A deep learning perspective," in *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, vol. 2, 2024, pp. 1–6. DOI: 10.1109/IATMSI60426.2024.10502843.

[39] A. I. Jony and A. K. B. Arnob, "A long short-term memory based approach for detecting cyber attacks in IoT using CIC-IoT2023 dataset," *J. Edge Comp.*, vol. 3, no. 1, pp. 28–42, May 2024.

[40] https://www.researchgate.net/publication/354114457_DETECTING_DDOS_ATTACKS_USING_LOGISTIC_REGRESSION, Accessed: 2024-5-21.

[41] *Amazon 'thwarts largest ever DDoS cyber-attack' — bbc.com*, https://www.bbc.com/news/technology-53093611, [Accessed 21-05-2024].

[42] *B4: Experience with a globally-deployed software defined wan: ACM SIGCOMM Computer Communication Review: Vol 43, No 4 — dl.acm.org*, https://dl.acm.org/doi/10.1145/2534169.2486019, [Accessed 21-05-2024].

[43] *Classification based on decision tree algorithm for machine learning*, en, https://www.jastt.org/index.php/jasttpath/article/view/65/24, Accessed: 2024-5-21.

[44] *Google Cloud mitigated largest DDoS attack, peaking above 398 million rps — Google Cloud Blog — cloud.google.com*, https://cloud.google.com/blog/products/identity-security/google-cloud-mitigated-largest-ddos-attack-peaking-above-398-million-rps, [Accessed 21-05-2024].

[45] S. Kottler, *February 28th DDoS Incident Report — github.blog*, https://github.blog/2018-03-01-ddos-incident-report/, [Accessed 21-05-2024].

[46] *No title*, https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/, Accessed: 2024-5-21.