FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING
DEGREE PROGRAMME IN ELECTRONICS AND COMMUNICATIONS ENGINEERING

MASTERS THESIS

Generation of a dataset for network intrusion detection in a real 5G environment

Author	Samarakoon Mudiyanselage Sehan Madhuka Bandara Samarakoon
Supervisors	Dr. Pawani Porambage (UO) Dr. Maheshi B. Dissanayake (UoP)
Second examiner	Prof. Mika Ylianttila
Technical supervisors	Dr. Madhushanka Liyanage Yushan Siriwardhane

July 2022

Samarakoon, Sehan (2022) generation of a dataset for network intrusion detection in a real 5G environment Faculty of Information Technology and Electrical Engineering, Degree Programme in Electronics and Communications Engineering, 89 pages.

ABSTRACT

As 5G technology is widely implemented on a global scale, both the complexity of networks and the amount of data created have exploded. Future mobile networks will incorporate artificial intelligence as a crucial enabler for intelligent wireless communications, closed-loop network optimization, and big data analytics. In these future mobile networks, network security would be of the utmost importance, as many applications expect a higher level of network security from the networking infrastructure. Therefore, conventional procedures in which action is taken following the detection of an attack would be insufficient, and self-adaptive intelligent security systems would be required. This paves the door for AI-based network security strategies in the future. In AI-based security research, the lack of comprehensive, valid datasets is a persistent issue. Publicly accessible data sets are either obsolete or insufficient for 5G security research. In addition, mobile network providers are hesitant to share actual network datasets due to privacy issues. Hence, a genuine data set from a real network is extremely beneficial to AI-based network security research. This study will describe the creation of a genuine dataset containing several attack scenarios implemented on a real 5G network with real mobile users. Since a fully operational 5G network is utilized to generate the data, this dataset is characterized by its close resemblance to real-world situations. In addition, data is collected from multiple base stations and made available as independent datasets for federated learning-based research to build a global model of intelligence for the entire network. The obtained data will be processed to identify the optimal features, and the accuracy of intrusion detection will be validated using several common machine learning and neural network models such as Decision Tree, Random Forest, K-Nearest Neighbor, Support Vector Machines and Multi Layer Perceptron. A detailed analysis of a binary classification to detect malicious and non-malicious flows as well as a multi class classification to detect different attack types is presented.

Keywords: 5G, datasets, intrusion detection, AI, ML

CONTENTS

PREFACE

This thesis titled 'Generation of a dataset for network intrusion detection in a real 5G environment' is an original work by myself submitted for the Masters Double Degree in Wireless Communication Engineering at University of Oulu. I would like to assure that this thesis, or a thesis substantially comparable to it, has not been submitted to any other degree program. All of the work presented in this thesis was conducted at the Centre for Wireless Communication Engineering at University of Oulu under the main supervision of Dr. Pawani Porambage.

I am extremely grateful to my supervisors Dr. Pawani Porambage from University of Oulu and Dr. Maheshi B. Dissanayake from the University of Peradeniya for their endless support and guidance during this research. Also, I would like to extend my gratitude to Prof. Mika Ylianttila for providing all the resources and financial support for the project. In addition, I would also like to extend my gratitude to Dr. Madushanka Liyanage for the continuous support and assistance throughout the thesis process. I would also like to express my heartfelt gratitude to Yushan Siriwardhana for the continuous guidance and suggestions given to me at all times. Finally, I take this opportunity to thank my family and friends for the encouragement and support.

Oulu, July, 2022

Sehan Samarakoon

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
GPS	Global Positioning System
IP	Internet Protocol
eMBB	Enhanced Mobile Broadband
uRLLC	Ultra Reliable and Low Latency Communication
mMTC	Massive Machine Type Communictions
VR	Virtual Reality
IoT	Internet of Things
SDN	Software Defined Networks
NFV	Network Function Virtualization
MEC	Multi-access Edge Computing
RAN	Random Access Network
DoS	Denial of Service
DDoS	Distributed Denial of Service
LOIC	Low Orbit Ion Canon
HULK	HTTP Unbearable Load King
IDS	Intrusion Detection Systems
SIDS	Signature Intrusion Detection Systems
AIDS	Anomaly-based Intrusion Detection Systems
PCA	Principal Component Analysis
SVD	Single Value Decompsition
KNN	K-Nearest Neighbors
SVM	Support Vector Machines
RBF	Radial Basis Function
ANN	Artificial Neural Networks
MLP	Multi Layer Perceptron
FTP	File Transfer Protocol
DNS	Domain Name System
SSH	Secure Socket Shell
HTTP	Hyper-Text Transfer Protocol
SMTP	Simple Mail Transfer Protocol
IMAP	Internet Message Access Protocol
POP3	Post Office Protocol
HTTPS	Hyper-Text Transfer Protocol Secure
VM	Virtual Machines
TCP	Transmission Control Protocol
SYN	An abbreviation for synchronization
FIN	An abbreviation for finish
UDP	User Datagram Protocol
RST	An abbreviation for Reset
ACK	An abbreviation for Acknowledgement
ICMP	Internet Control Message Protocol
LDoS	Low Rate DoS
GTP	GPRS Tunneling Protocol
MPLS	Multi Protocol Label Switching
GRE	Generic Routing Encapsulation
MTU	Maximum Transmission Unit
BS	Base Station

LIST OF SYMBOLS AND ABBREVIATIONS

eNB	E-UTRAN Node B / Evolved Node B
5GTNF	5G Test Network Finland
SFTP	Secure File Transfer Protocol
GSM	Global System for Mobile Communications
UMTS	Universal Mobile Telecommunication System
LTE	Long-Term Evolution
5G NR	5G New Radio
GTP-U	GPRS Tunneling Protocol - User
GTP-C	GPRS Tunneling Protocol - Control
GTP'	GPRS Tunneling Protocol - Prime
CDF	Charging Data Function
CGF	Charging Gateway Function
PCC	Pearson Correlation Coefficient
ROC	Receiver Operating Characteristic
AUC	Area Under the ROC Curve
TP	True Positive
FP	False Positive
FN	False Negative
TN	True Negative
DT	Decision Tree
RF	Random Forest

1 INTRODUCTION

Mobile communication networks have advanced fast over the last three decades to reach where they are now. Second-generation networks, which were introduced in the 1990s, have gained worldwide recognition as a paradigm shift that ushered in a new age in digital communications [1]. Since then, as the requirements of people as well as businesses increased over the years, the communication requirements has also increased rapidly. For any mobile network, higher data speeds, lower latency, and greater reliability have become essential. To address these issues, 3G and 4G technologies were invented later, each with its own set of features.

The third generation of wireless communication systems, which emerged in 1998, was designed to deliver high-speed internet while maintaining stable connections over large distances [2]. By using packet switching instead of circuit switching, the technology was able to give significant performance enhancements over 2G systems. The higher speeds facilitated in ability to stream radio and television on 3G devices [2]. Furthermore, the technology provided value added services such as GPS and video conferencing [3].

Verizon launched the very first 4G network in the United States in 2011, promising a 10-fold increase in speed over the previous 3G network [2]. In comparison to 3G networks, 4G technology totally eliminated circuit switching by using IP (Internet Protocol) even for voice data, while delivering higher bandwidth and more features [2]. The goal of 4G technology was to deliver high-quality audio/video streaming over an end-to-end IP network [3]. The advancement of technology caused a reconsideration of how smartphones were viewed, leading them to be viewed as computers of modern age.

The rapid growth in the number of wirelessly linked devices, together with the development of new applications, cleared the way for the need for even faster speeds than 4G. This, combined with the need for enhanced mobile broadband (eMBB), ultra-reliable and low-latency communication (uRLLC) and massive machine type communications (mMTC) has resulted in 5G emerging as a technology capable of handling numerous network functions and use cases. Along with higher speeds, 5G networks are aimed at providing improved latency, lower energy consumption, as well as reduced costs [2]. The 5G network is also aimed towards addressing the challenges such as varying Quality of Service (QoS) requirements, interoperability of different wireless devices and interfaces and average spectral efficiency [4]. Unlike many previous advancements, 5G has extended mobile communication from humans to things, as well as from consumers to verticals [5]. Not limiting to traditional mobile broadband services, 5G extends to applications such as Industry 4.0, Virtual reality (VR), Internet of Things (IoT) etc through the use of techniques such as Network Slicing, Software Defined Networks (SDN), Network Function Virtualization (NFV) and concepts such as Multiaccess Edge Computing (MEC).

As 5G communication technology is deployed more widely around the world, the number of devices and data being transmitted, as well as the variety of use cases, will result in significantly more complicated networks in the near future. Furthermore, beyond 5G networks are expected to include connected intelligence [6], necessitating lower latency and faster response times for effective network management. In order to attain these connected intelligence's, future networks will be heavily relied on Artificial Intelligence (AI) and other supporting technologies such as big data analytics, closed-loop network optimization as well as advances in wireless communication itself.

Because of the greater capacity of the networks, new applications in the beyond 5G era demand some level of security from the networking infrastructure as well. The security features of these networks must be carefully implemented as they become more sophisticated and diverse technologies are merged. Especially, the security challenge with 5G networks is significant since it is not only associated with the wireless aspect of the technology, but also with associating technologies that are fundamental to 5G [7]. The large number of IoT devices and provisions

for applications like smart homes, smart grids, transportation, and healthcare offers a significant challenge when it comes to developing security solutions for 5G [8].

1.1 Background and Motivation

In future networks, a high level of attack scenarios are expected especially due to the leverage of AI. Adversaries are capable of learning the network through use of AI to plan and carry out intelligent attacks against the network. In such circumstances, conventional security mechanisms which react only after an attack has been detected may not be the most efficient form of defense. The future networks will need a system that can detect assaults early on and prevent them from compromising the network performance. Therefore, a system that can deploy proactive, self-adaptive and self intelligent securing measures is a definite requirement for beyond 5G networks [8]. Proactive security schemes can ensure the mitigation of potential security flaws in the network through the use of intelligence gathering [8]. In modern high-speed 5G networks which assures high levels of latency, these security mechanisms should also guarantee that the network is not subjected to bottlenecks. Therefore, the detection and mitigation techniques used should be near real-time at all instances.

At this point, artificial intelligence (AI) and machine learning (ML) approaches may be able to aid in delivering the above-mentioned intelligence in order to create proactive security systems. The use of AI/ML algorithms in other domains indicate that it can be effectively applied for network security as well. It can be used to build a secure network that can detect threats and offer solutions in real time. AI and ML algorithms utilize massive volumes of data to identify common patterns for intrusion detection. In this context, datasets are crucial to the effective application of AI and ML-based security systems. The limited availability of datasets that corresponds to modern networks has been a primary concern in security research. Specifically, despite the fact that 5G networks are already been deployed globally, the non availability of a public dataset which resembles the network flows in 5G is a significant issue. As a communication technology that will define the next decade in telecommunications, a dataset to validate effective AI based security measures would be a significant contribution.

1.2 Research Problem

The quality and quantity of data has a significant impact on the success of any ML/AI based solution. A significant amount of data is always expected to be supplied into a machine learning model in order for the model to train properly and identify sufficient patterns in the data. Due to a lack of data, the system may record a large number of inaccurate predictions, jeopardizing the system's accuracy. In a similar vein, determining the applicability of a number of different machine learning or deep learning models relies heavily on the quality of the data.

The ongoing research work in the context of AI based security solutions for 5G networks have suffered due to lack of high quality training dataset to evaluate their solutions. For the most part, researchers have chosen publicly available datasets that are out of date or limited in applicability to 5G to test their security solutions. Furthermore, due to privacy concerns and other legal barriers, mobile network operators have been hesitant to publish data from their networks. Another reason for the reluctance could be the risk of exposing network vulnerabilities to third parties. As a result, the AI based security implementations and research are been progressing behind.

Therefore, developing a high-quality 5G dataset with a variety of attack types and sufficient

data is vital for the entire research community. The end goal of this project is to create a 5G intrusion detection dataset using a real-world 5G scenario with varying attack types.

1.3 Selected scope

Due to the multitude of applications supporting 5G, the networks can be far more complex than in the previous generations. The incorporation of additional supporting technologies, such as SDN, NFV, and concepts such as MEC, improves latency but also introduces new vulnerabilities in the network.

This study centered on a plausible 5G network with Multiaccess Edge Computing (MEC) capabilities. A MEC is designed to extend cloud capabilities of the network towards the edge. This is achieved by deploying storage and processing resources near the RAN's edge, which allows some cloud capabilities to be moved closer to the network, hence reducing latency [9]. In our study, having a MEC allowed us to create distinct networks for the attacker and the target while still allowing for sufficient attack variants.

Out of many possible attack scenarios, different types of DoS/DDoS attacks and port scans which are frequent in today's networks were carried out. Previous research efforts and the capabilities of our test network were taken into consideration while selecting the types of attacks to use. The attacks were chosen such that multiple characteristics are covered in terms of protocols, rate of packets, duration of attack etc.

1.4 Methodology

As the underlying infrastructure for this experiment, the 5G test network (5GTN) at the University of Oulu was utilized. Mobile devices, 2 base stations, firewalls, switches, 5G modems, and personal computers were all part of the network that was put in place to make a modern 5G wireless communication system possible. These components combined with MEC capabilities ensured that the network closely resemble the network architectures of the modern age. The attacks were carried out through Raspberry Pi's connected to the 5G modems using Wifi. The modems were connected to the Pico base stations. The victim was considered as a linux machine at the MEC. Both the attacking network and the network belonging to the victim were built in such a way that they were isolated from one another.

Tools such as Nmap, Hping3, Goldeneye, LOIC, HULK, Slowloris, and Torshammer were utilized in the execution of various sorts of port scanning as well as DoS and DDoS attacks. The availability of benign traffic was ensured at all times during the attacks by generating it using actual mobile devices linked to each of the base stations. Data collection was carried out in two days from both the Pico base stations in the network.

After the data collection, the analysis and data preprocessing was done through different tools. The argus tool was used to generate the network flows from the packets. The network flows in the dataset were labelled to ensure they can be used for supervised ML models. Also, in order to feed the data into ML models, a feature selection had to be carried out to eliminate irrelevant features from the dataset. The top most features were fed into different ML models to examine their suitability for intrusion detection in this dataset. The best model was picked based on the levels of accuracy as well as the training time of the model.

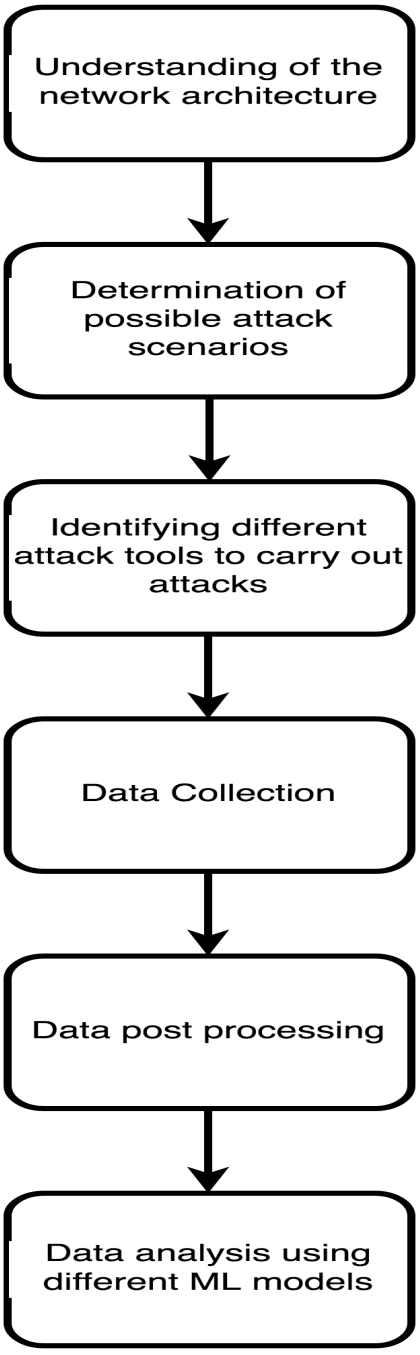


Figure 1.1: Overview of the Methodology

1.5 Contribution

Since 1998 up till the present day, a variety of datasets have been made accessible for the purpose of network intrusion detection. The vast majority of these datasets were produced with the assistance of virtualized networks or networks that had been constructed with the explicit

purpose of producing the dataset. In contrast, this dataset was constructed on a functional 5G test network, which creates an environment that is quite similar to a real-world scenario. In addition, the previous efforts have concentrated on the development of benign traffic through the use of a variety of tools and simulators. On the other hand, this work made use of actual mobile users within the coverage area of base stations in order to generate the benign traffic.

In addition to that, it can be claimed that this is the first dataset that is now accessible for 5G. The accuracy and effectiveness of machine learning-based intrusion detection is strongly dependent on the quality of the data. Complete network configurations, a variety of attacks, anonymity, heterogeneity, complete traffic, labeling, complete interaction, metadata, complete capture, and feature sets are some of the characteristics that are considered to be essential for a comprehensive and valid dataset for the purposes of network intrusion detection [10]. Despite the fact that the majority of contemporary datasets have been able to address these criteria, the behavior and traffic characteristics of 5G networks are very different from the testbeds and simulation platforms that were used to collect these datasets. As a consequence of this, machine learning-based network intrusion detection solutions have been implemented with datasets that are either limited or irrelevant. This is a significant obstacle in the path of research on machine learning-based intrusion detection.

Furthermore, one of the things that sets this dataset apart from others is the fact that data has been collected from both of the Pico base stations. This may prove valuable in the development of federated learning-based systems for the detection of network intrusions. Through the use of it, one can realize a global intelligent security model for the entirety of the network. Therefore, the overall contribution of this work is to provide a comprehensive, valid, and up-to-date dataset for the research community in beyond 5G networks. This dataset will eventually assist in the achievement of intelligent and self-adaptive networks in the future, which is the ultimate goal of this work.

1.6 Organization of the thesis

The thesis contains six chapters in total with the first chapter giving a descriptive introduction to the research. Chapter 2 provides a detailed litereature review on the existing datasets and their shortcomings that can be addressed. Further, the section provides a review on the machine learning and supervised learning specifically in intrusion detection of netwprks. In addition a detailed idea about all the attacks performed as well as the tools used are presented. The Chapter 3 focuses on the data collection phase of the work. A detailed description on the network architecture, attacks and data collection are provided. A descriptive discussion on the post data processing is provided in Chapter 4. This includes different steps from network flow creation to feature selection. The Chapter 5 presents the results for different number of features while Chapter 6 discusses the overall research and provide an analysis on overall results. The thesis is ended with a conclusion that explores potential future work in Chapter 7.

2 LITERATURE REVIEW

Since the initial attempt of building datasets for network intrusion detection in 1998, numerous studies have been conducted and published. Throughout the years, datasets that are been published have undergone several advancements in the variety of attacks, network configurations, volume of traffic, types of protocols, etc. However, these datasets still contain certain limitations and have failed to adequately represent the modern networks and the associated technologies. This chapter will provide a comprehensive overview of existing datasets and their limitations, as well as a thorough analysis of some common machine learning algorithms and their application to intrusion detection especially in modern 5G networks. In addition, the various kinds of attacks and tools utilized in the research are discussed.

2.1 Machine Learning and Supervised Learning in IDS

Recent improvements in 5G and beyond networks have led to its integration with a variety of other technologies to give users the greatest experience possible in terms of speed, dependability, and cost. The ever-increasing number of users and the number of devices resulting through different use cases of 5G such as the Internet of Things (IoT), combined with these needs have caused networks to become rather massive and significantly more complicated. This increased complexity and data traffic have led to the emergence of new vulnerabilities, threats, and attacks in modern 5G networks. In addition, the complex architecture and high latency requirements has made it more difficult to efficiently detect network security risks. In addition, the increased bandwidth, spectrum utilization, and data rates of 5G networks have shifted the security and privacy landscape from the individual device to the service provider network [11]. In such scenarios, the network should be intelligent enough to detect vulnerabilities and address them in real time.

Therefore, network intrusion detection systems, also known as IDS, are essential for any network because they have the capacity to withstand attacks from third parties outside the network, whereas a firewall might not be able to accomplish this duty [12]. In the context of 5G networks, it is anticipated that it would accommodate a far higher level of heterogeneity than the previous generations. For example, 5G networks support vehicle to vehicle communications, smart homes, smart buildings, and smart cities. Moreover, the 5G network architecture for the Internet of Things (IoT) will necessitate more robust and adaptive approaches to handle the primary network and device-side security flaws [11]. Due to both external and local intrusions, the security of these networks will become significantly more challenging. As a result, the network infrastructure should be equipped with a higher level of security to survive these threats.

In the wider context, these intrusion detection systems (IDS) can be split into two categories based on their method of detection, which can be signature-based or anomaly-based. Pattern matching is the foundation of signature intrusion detection systems (SIDS) and these systems are designed to identify and neutralize known threats [13]. They are also sometimes referred as knowledge-based detection or misuse detection. In these systems, an alarm signal is triggered when an intrusion coincides with a signature that already exists in the signature database [14]. The disadvantage of this type of intrusion detection system is that it will have difficulty detecting zero-day attacks because there will be no matching signature in the database until the signature of the new attack is extracted and stored [14]. In the age of 5G networks, techniques based on artificial intelligence are employed to study the network and then execute unique and sophisticated attacks. Signature-based intrusion detection systems are insufficient to detect

these types of attacks. . Additionally, conventional SIDS identify attacks by inspecting network packets and comparing them with signatures in the database. However, they are unable to recognize attacks that spans across multiple number of packets at a time. Therefore, SIDS will be inadequate due to the ever-increasing frequency of zero-day attacks that may employ AI techniques to learn the network and plan attacks accordingly in 5G networks.

It has been identified that the potential solution to address these limitations in SIDS is the use of Anomaly based Intrusion Detection Systems (AIDS). Machine learning, statistical based or knowledge based methods are used in AIDS to make the system learn the normal behavior of traffic. Then, any variation from this typical trend in the observed flow of traffic will be considered as an intrusion [14]. Machine learning in particular has shown to be particularly helpful in this area because it has the ability to learn from a huge number of data and then apply the learnt output to recognize deviations from the behaviors that are considered normal. In the era of 5G networks, the generation of large amounts of data in the network and the computational ability of processing such data enables the efficient use of AI/ML techniques for intrusion detection. In a fast phased and latency intolerant network environment, ML and AI techniques could be used to simulate resilient and dynamic security algorithms that can help detect network intrusions and provide potential solutions in real time.

The use of ML in intrusion detection has the advantage of allowing systems to learn and enhance their automatic capacity from experience without the need for programming specifically [15]. Also, it is able to function accurately to detect attacks in vast amounts of data in a shorter period of time, which is a significant advantage. Under ML too, a wide variety of approaches are available for use in the process of intrusion detection. Learning can be primarily broken down into three categories: supervised learning, unsupervised learning, and semi-supervised learning.

For the purpose of determining a relationship with the data, supervised learning approaches make use of classes that are completely labeled. The classification process has two phases namely, training phase and the testing phase. In the training phase, a dataset is fed into the model, which is then utilized to determine normal and malicious behavior patterns and features. In the testing phase, the system is fed an entirely new set of data to determine the accuracy of the predictions done based on previously trained model. Logistic Regression, Decision Trees, Random Forest, Nearest Neighbor, Naive Bayes, and Neural Networks are some of the most common forms of supervised learning algorithms that are accessible for classification. These many models can be utilized in accordance with the various kinds of data that are available in order to attain the best possible prediction accuracy or other metrics.

Unsupervised learning differs from supervised learning in that there are no labeled data. This method attempts to discover the hidden structure inside unlabeled data to determine the intrusions. Therefore, no training data exists for unsupervised learning. Methods of unsupervised learning could be effective in situations in which labeling attack data is difficult or even impossible to do [16]. Unsupervised learning can be carried out either through clustering techniques or dimensionality reduction techniques [15]. K-means, K-mediods and C-means are some of the popular clustering based methods available. The Principal Component Analysis (PCA) and Single Value Decomposition (SVD), among others are both examples of dimension reduction methods.

Semi-supervised learning is theoretically positioned between supervised and unsupervised learning, and it enables the use of huge quantities of unlabeled data available in the majority of use cases in conjunction with generally smaller amounts of labeled data [17]. Also, it should be noted that it can be used vice versa too as these semi-supervised algorithms attempts to improve the performance of supervised or unsupervised learning by making use of relevant data that is usually available with the other.

Among these different ML techniques, each one of them has their advantages according to the type of data available and the application. However, in the case of intrusion detection, supervised learning-based algorithms have been more prevalent. Using supervised learning to detect network intrusions is largely reliant on the quality of the available data. The quality can be attributed to the correct labeling of the data, as it plays a crucial part in supervised learning algorithms. The correct labeling of malicious and non-malicious data, as well as other forms of attacks, is extremely challenging in some networks. Therefore, a dataset with proper labelling and sufficient amounts of data is a scarce resource for most researches.

During the training phase of supervised learning, the relevant features and classes are discovered, and the algorithm learns from the data samples that are provided [14]. In order to shorten the time required for the training process, it is possible to feed the model with fewer features while maintaining an acceptable degree of accuracy. This technique, known as feature selection or feature reduction is capable of determining the attributes that have the most impact on the target variable. These attributes can then be used to discover the underlying relationship with the target variable. In the testing phase of supervised learning, a previously unknown set of data is fed into the model so that it predicts the outcome based on previously acquired knowledge.

Different kinds of supervised learning algorithms have been used in literature for intrusion detection purposes. The learning algorithm must be able to predict the accuracy of any combination of unknown data at a level that is acceptable. Hence, the classification algorithm should have the ability for reasonable generalizations [14]. The Decision Tree algorithm is a basic supervised learning technique that is available. It is applicable to both classification and regression issues. The categorization in decision tree classification is based on rules. Decision trees consist of three components namely decision nodes, branches and leaves. A test attribute is recognized at the decision node. Then, each branch represents a potential decision based on the test attribute's value. The leaves indicate the class to which an instance belongs [18].

Another supervised learning model is the Random forest. This can be considered as an extension of decision trees as it contains multiple decision trees and is based on ensemble learning. The final classification result is determined by a vote from all the decision trees. Random forest can address the over-fitting issue of decision trees and has a high tolerance for noise and outlier data [19].

The K-nearest neighbors is also quite a popular non-parametric classifier used in supervised learning. The objective of K-nearest neighbors is to assign unlabeled samples requiring categorization to the class of their nearest neighbors. The number of neighbors to be considered can be specified. KNN can be applied as a benchmark for all other classifiers as it generally provides good performance in IDS [20].

The Support Vector Machines algorithms works to identify a hyperplane in N-dimensional space which classifies the data distinctly. SVM does this by maximizing the distance between the data points in the classes and the hyperplane. SVM'a are said to be valuable when a large number of attributes are present with lesser amounts of data points [14]. Also, they use kernel function to map data into higher dimensional space and different types of kernel functions such as linear, Radial Basis Function (RBF), polynomial can be used. SVM can also be used for multi class classification as seen from the literature.

Apart from these machine learning models, neural networks can also be used to increase the accuracy's further using different numbers of hidden layers and neurons. Artificial Neural Networks (ANN) are inspired by the human brain and can be used to solve complex classification problems. They are usually made up of a number of highly interconnected processing components called neurons that collaborate to resolve complicated problems [21]. A Multi Layer Perceptron (MLP) is a type of fully connected feed-forward ANN that is widely used in IDS as one of the most adaptable and powerful classification methods [22]. By manipulating the design

of an MLP, a successful classification with increased speed and precision can be achieved [23]. This is often accomplished by adjusting the number of hidden layers and the number of neurons in each layer until the optimal configuration is found.

2.2 Existing Datasets

The datasets are crucial for assessing the performance of intrusion detection systems. There have been numerous datasets available for this purpose under various network architectures and conditions over the years. As the capabilities of attacks and corresponding technology grew, new datasets were developed to remedy flaws in existing ones. Following is a discussion of popular publicly accessible datasets, their features and limitations.

The DARPA dataset is considered the first publicly accessible dataset for intrusion detection purposes. At the time, the dataset introduced by the MIT Lincoln laboratory was an enormous contribution to the IDS research community. The DARPA 1998 and DARPA 1999 datasets were generated with attacks such as port scans, denial of service, buffer overflow, and rootkits and contained seven and five weeks of network traffic, respectively [24]. Although this dataset was widely utilized for research work, it was heavily criticized for ignoring real-world circumstances [25]. The criticisms are mostly due to the insertion of artificial traffic, high quantities of redundancy, absence of false positive occurrences, and abnormalities in attack data [26][27]. In addition, it is deemed obsolete for evaluating IDS on modern networks in terms of both the variety of attacks and the network infrastructure [10].

To address some of the problems involved with the DARPA datasets, the tcpdump component of the DARPA dataset was utilized to generate the KDD CUP 99 dataset. However, the majority of deficiencies associated with the DARPA dataset were also present in this dataset [10]. KDD CUP 99 includes over 20 distinct kinds of attacks, such as DoS and buffer overflows, and an explicit test and training subsets. The datset is not accessible in either the standard packet format or the flow-based format. This dataset is critiqued for holding a huge number of redundant and duplicate instances, and as a result, it is riddled with data corruptions that have contributed to distorted conclusions [10]. Nevertheless, despite its flaws, it is still widely utilized by the research community.

The Shmoo group created the DEFCON dataset in 2000, which included DoS and buffer overflow attacks. Subsequently, the DEFCON-8 dataset was also made available in 2002, along with a number of additional attacks, such as port scans, sweeps, FTP, and telnet protocol attacks [10]. This data collection differed from the previous ones because it featured information gathered during a "Capture the flag" competition [27]. This strategy faltered, as the production of network packets during a competition is significantly different from network traffic in general. Due to the nature of the competition, this contains very high amount of attack traffic as opposed to normal traffic.

There are also a variety of datasets that have been released by CAIDA. This primarily consists of the CAIDA OC48 dataset, which is data observed on an OC48 link in San Jose, the CAIDA DDoS attack dataset, which covers network traffic during a one-hour DDoS attack, and the CAIDA Internet trace released in 2016, which includes passive traffic traces from CAIDA's equinix-chicago monitor on high speed internet backbone [10]. Reportedly, the CAIDA records contain instances that are unique to a certain type of attack or internet activity [27]. In addition, the payload, protocol details, and destination of these datasets were anonymised.

The LBNL dataset that was made public by Lawrence Berkeley National Laboratory includes 100 hours of network traffic with traces of entire packet headers captured from their enterprise networks [10][27]. The dataset has been subjected to a significant amount of anonymization,

particularly with regard to packet headers and IP addresses [28].

The CDX dataset that was developed by the United States military put forward the idea of constructing an intrusion detection dataset through the use of network warfare competitions. The recorded traffic from the network warfare competition that took place over the course of four days is available in a packet based format [29]. In order to carry out the attack, the attackers made use of various attacking tools, such as Nikto and Nessus. The capture also involved benign traffic in the form of web, email and DNS lookups [29]. The absence of diverse range of attacks and the lower volume of data are identified as major concerns in this dataset.

Datasets were published by Kyoto University from 2006 onwards with data created through honeypots. This restricts the attack scope to only those attacks that can be directed at honeypots. Normal traffic in this dataset was derived from DNS and email traffic, which may not accurately represent the traffic in an actual network [10]. This dataset was converted from packet-based data to a new format termed sessions using the IDS Bro tool in which each session comprised of 24 attributes [30]. The dataset contains a vast amount of information because it was collected over a three-year period. However, the non representation of common normal traffic protocols and the limitations of the attacks is identified as drawbacks.

In 2009, the University of Twente introduced a dataset also based on honeypots that offer web, FTP, and SSH services. This dataset likewise suffered the same fate as the Kyoto dataset because it only comprises network traffic that was collected using honeypots. As a result, practically all of the flows in this dataset are malicious, and there are very few instances of normal traffic [31]. The dataset has labels and is realistic; nonetheless, there is a lack of a variety of attacks and a significant amount of traffic, which is been regarded as concerns.

The ISCX2012 dataset introduced by University of New Brunswick contained attacks such as DoS, DDoS, Brute force, Infiltration attacks. The dataset was created by carrying out traffic capture in an emulated network over a period of one week. It was generated through a dynamic approach with alpha profiles representing attack scenarios and beta profiles with normal background traffic. The resulting dataset was generatd in a packet based format as well as bidirectional flow based formats [32]. The network traffic comprised of protocols such as HTTP, SMTP, SSH, IMAP, POP3, and FTP, but modern protocols such as HTTPS were absent [10].

The AFDA dataset from University of New South Wales and Australian Defense Force Academy consists of two datasets namely ADFA-LD and ADFA-WD. The two separate data collections was done through Linux and Windows operating systems. Since the dataset contains information on zero-day assaults, it is ideal for illustrating the distinctions between SIDS and AIDS. The typical behavior of a user was demonstrated using a variety of different tasks, ranging from browsing of websites to preparing documents in LATEX [14]. The dataset also included system calls for different types of attacks. The dataset is accused of lack of attack diversity and variety of attacks. In addition some attacks in this dataset are considered to be hardly separable from the normal behavior [33].

A more recent dataset: UNSW-NB15 dataset contains data that includes both regular and malicious traffic in a packet-based format. This data was produced using the IXIA Perfect Storm program in a smaller emulated environment over the course of 31 hours. The data includes a variety of attacks, including denial-of-service, exploits, fuzzers, backdoors, and worms among others. This dataset is also accessible in a flow-based format, complete with predefined test train splits, and is publicly available [34]. However, the dataset was created using a simulated or artificial environment to carry out different types of attacks [35].

The Canadian Institute of Cyber Security has published multiple network intrusion detection datasets. The CICDS 2017 dataset includes many sorts of attacks, including Brute Force FTP and SSH, DoS, DDoS, HeartBleed, and Infiltration attacks [36]. The dataset is labeled with the

use of five primary characteristics, including timestamp, source and destination IP addresses, and source and destination ports [14]. With a network consisting of servers with different operating systems and involvement of network components such as modems, routers, firewalls and switches, the networking infrastructure can be considered as realistic. Non-malicious background traffic was generated using a benign traffic profile that was collected from a number of real users. In a real network, the inherent intricacy of this type of data from a concept such as profiling could be a challenge. The same institution has produced another dataset with different types of DoS attacks named as CIC DoS. In this dataset, preceding the CICDS 2017, the authors have focused on the application layer DoS attacks [37]. 8 different DoS attacks have been carried out on a similar network as in the previous work. The attack free traffic from the ISCX 2012 dataset has been used to form the benign traffic and attack data has been combined only after the data collection [32]. This has reduced the realistic nature of the dataset as timestamps may differ.

The BoT-IoT dataset introduced in 2019 tries to address the non availability of datasets with IoT traffic. In this work different types of attack scenarios including probing attacks, DoS and information theft have been conducted. The network comprised of virtual machines (VM) and additional firewalls and IoT services were simulated through the Node-red tool. The Ostinato tool was responsible for the generation of benign traffic artificially [35].

The datasets generated over the years has addressed certain issues related to attack types and formation of benign traffic in their work. However, none of them have addressed the specificity of traffic generated in a 5G network along with its associated technologies. In fact none of the datasets has accompanied mobile phones in their work. This sets our dataset apart from other datasets as we will be using mobile phones to generate live traffic including different protocols. In an age where mobile phones, laptops, IoT devices, machines etc. are interconnected through networks, this produces a realistic network architecture of a modern network. Additionally, in the past, the production of benign traffic has been inconsistent, with some datasets introducing benign traffic after data collection. Our focus was to generate live traffic in the network in real time to represent a realstic network even in a small scale. Moreover, as can be seen from the descriptions of previous datasets, the use of emulated environments for data collection has been a common occurrence throughout. In contrast our work comprises of a functional 5G test network which signifies the validity of it further. Further, our work will provide the dataset in packet-based format and flow-based format as well. The existence if data from multiple points of the network is also a unique feature in our dataset. With this availability, one can realize federated learning based security solutions using our dataset. In this way, the non availability of modern datasets has hindered and limited the research progress in ML based intrusion detection systems for 5G networks and we hope that this dataset will be able to address it.

2.3 Attacks

The dispersed and heterogeneous nature of 5G networks has increased the number of potential attack scenarios significantly. Specifically, the 5G HetNets threat surface includes threats such as denial of service (DoS), malware propagation, and malicious port scanning among others [38]. The integration of different radio technologies as well as associated technologies in 5G HetNets has resulted in further increase of the impact of these attacks. Hence, port scanning and Denial of Service (DoS), which can be prominent in different 5G network configurations, are the focus of this work.

In a typical 5G HetNet, attackers inside or outside the network can perform such attacks and due to the distributed nature of users, the detection may need to be performed at different

locations [38]. To address this need, we perform the attacks from two separately isolated networks to a target in a completely isolated network.

2.3.1 Port Scans

Port scan is a process typically carried out before performing attacks, in order to identify the open ports and attack opportunities present in a vulnerable host. This is carried out through observing the response of each port in a host by sending out different types of requests. The status of the received response may even define the operating system of the target along with the open ports and other available services [39]. Majority of attacks are preceded by a scanning procedure such as port scanning before execution since it aids in identifying the exploitable hosts. An IDS that can detect port scans is usually advantageous, as they are a precursor to more severe attacks in the future. Therefore, it enables the system to recognize attacks early on and prevent more serious forms of attacks that may occur subsequently.

Different types of port scanning techniques exists which effectively performs the same task. However, the message requests that are sent differ in most of these techniques in terms of the protocols and flags. In a typical network, devices are connected with multiple other devices and services are maintained through different ports. Generally, there are 65536 defined ports in a device, which can be classified as well known ports (0-1023), registered ports (1024-49151) and private ports (49152-65535) [39]. Any port scan will consist of sending a message to each individual port and waiting for a response. The response usually defines the status of the port. TCP based port scans are common as it provides a good feedback for the attacker due to its connection oriented nature. UDP-based port scanning can also be observed; however, because it is connectionless, this type of scanning does not provide the attacker with sufficient information in most cases. Port scans can be mainly divided into five types based on their characteristics as shown in 2.1. The stealth scans involves sending of TCP packets to the destination with stealth flags such as SYN, FIN, NULL. SYN scan can be considered as a type of stealth scan. SOCKS is an Internet Protocol that enables a client and server to communicate with one another by exchanging information via a proxy server. An attacker who is doing a SOCKS port scan is able to access other hosts on the internet while concealing their true identity and location behind a proxy. Bounce scan exploits a vulnerability of the FTP protocol in order to scan into another device. A TCP scan is a hard to detect scan performed by sending requests for TCP three way handshake. Some of the TCP scans include TCP Connect(), reverse identification, IP header dump scan etc. In an UDP scan, UDP packets are sent to the related ports in order to observe the status of their response. In this work, three different types of port scans were performed namely SYN scan, TCP Connect scan and UDP scan.

2.3.1.1 SYN Scan

SYN scan can be described as the most prominent type of port scanning method available. This falls under the type of stealth scan where TCP packets are sent to the destination with stealth flags set. This scanning method can be considered speedy as it can scan thousands of ports in a matter of seconds.In addition to this, it may be used against any TCP stack that is compliant, as opposed to FIN/NULL/Xmas scans, which are dependent on the peculiarities of certain platforms.

During SYN scan, the attacker first sends a TCP packet with SYN flag set which exactly resembles the initiation of the three way handshake in any legitimate TCP connection. After

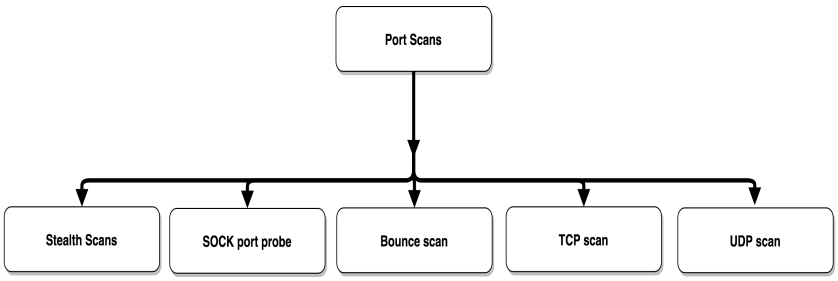


Figure 2.1: Main classification of Port Scans

this packet arrives to the targeted port, it sends back a response with SYN and ACK flags in the case of an open port. The attacker gets to know that the port is open from the response with SYN and ACK flags set. If the port is closed, the response would be a RST packet. Due to the fact that the attacker already possesses the necessary information from the first two packet flows between each other, the three-way handshake never gets completed in this kind of port scans. Therefore, it is sometimes referred to as half-open scanning. However, if the target has an open port, the attacker may send a RST paket in order to terminate the connection because, in the absence of this action, the target would continue re-sending packets that have the SYN and ACK flag set. In addition, if a port is unresponsive to a TCP initiation packet, this could be due to the presence of a firewall or the host being down. Figures 2.2 and 2.3 shows the packet flow during SYN scan for open and closed ports.

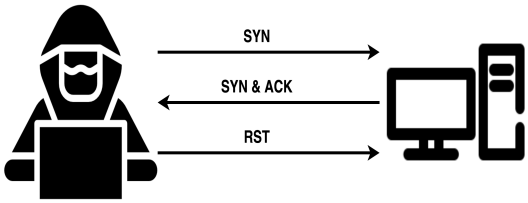


Figure 2.2: SYN scan of an open port

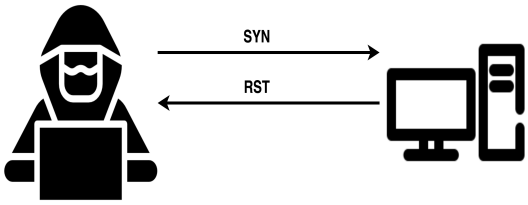


Figure 2.3: SYN scan of an closed port

2.3.1.2 TCP Connect Scan

TCP connect scan belongs to the TCP scan type in the initial classification done on Figure 2.1. This takes a similar approach to the SYN scan but the complete TCP three way handshake is made. This results in a longer time for the port scanning process and also require more packets to acquire the same information as in SYN scan.

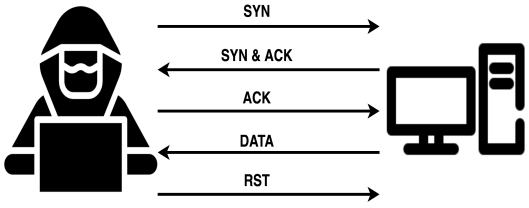


Figure 2.4: TCP connect scan of an open port

In the first two stages of a TCP connect scan, the same packet flows are anticipated as in a SYN scan (SYN and SYN ACK). However, in TCP connect scan the complete TCP connection is established in the subsequent phase without terminating the connection as with the SYN scan. This happens through an ACK sent by the attacker to the target as an acknowledgement for the SYN ACK that was sent. After the TCP connection is established, there exists possibility for the devices to exchange data as well. However, as soon as the attacker OS is aware that the connection is established, it terminates it through a RST packet as indicated in figure 2.4.

In the case of a closed port, the packet flows behave similar to the SYN scan in which an RST packet is sent from the closed port. Similarly, a firewall or host being down may indicate that the port is filtered.

2.3.1.3 UDP Scan

As the name suggests, an UDP scan involves sending UDP packets to the ports in the target to determine their response. For the majority of these ports, packets will be sent without a payload, with the exception of UDP ports, for which a protocol specific payload will be available. The response of the target or unresponsiveness is used to determine the status of the port. With UDP, a response from a desired port is highly unlikely, as an open port hardly responds to an empty packet. Due to this, the majority of ports may be in the *open | filtered state*, indicating that either the TCP/IP stack forwards empty packets to a listening application and discards them as invalid or a firewall is dropping packets without a response. This makes it difficult to determine whether a port is in open state exactly. However, the packets sent to a UDP port is likely to get an response as the packet payload is not empty. A response will be a clear indication of the *open* state. If the response is an ICMP port unreachable error, the port is determined to be closed.

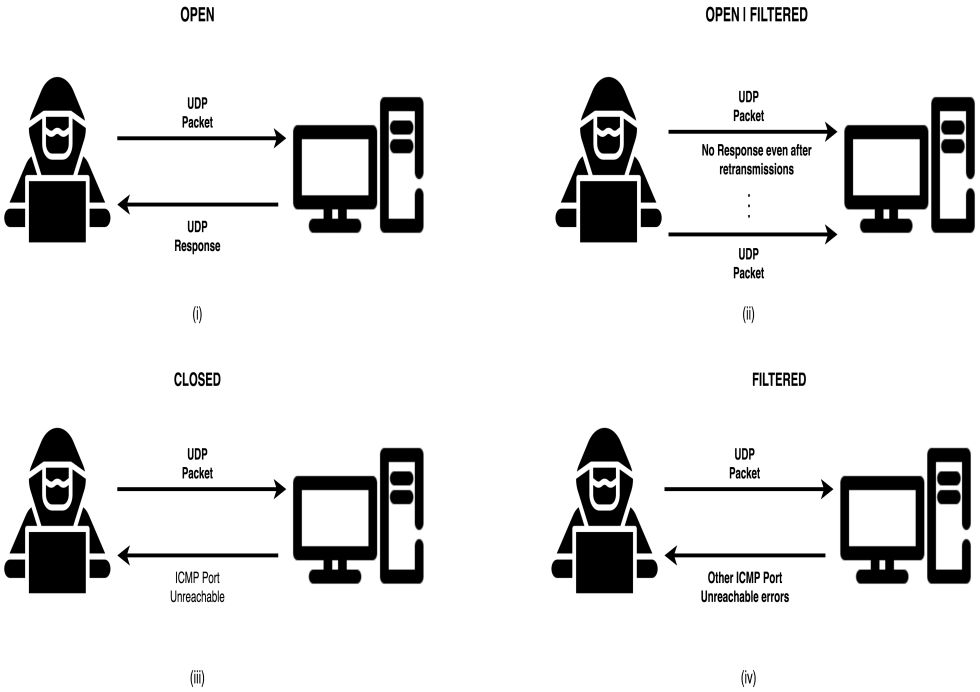


Figure 2.5: Instances of UDP Scan

2.3.2 DoS / DDoS Attacks

A Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack is an expected and common attack in any type of network configuration. Typically, a DoS attack is carried out to slow down or fully shut down a targeted server or a network, leaving it inaccessible to its intended users. The majority of the time, this is accomplished through the deployment of flooding techniques or transmitting of information that can trigger a crash. DDoS attacks are essentially the same type of attacks performed by multiple devices simultaneously. This could generate massive amounts of traffic on the network and eventually make it unreachable to other users. During a DDoS attack, the variety of network traffic originating from various network devices may vary. Traditional security methods may therefore be incapable of detecting these threats.

In the case of 5G network architecture, the DoS/DDoS attacks spans a large area in terms of the targeted resources. DoS attacks are designed to overwhelm physiscal and logical resources of target. In the context of 5G, there can be DoS attacks targetting the network infrastructure as well as the users or devices [40]. In this work, the focus will be on the DoS attacks targeted at users or devices. Since there are no security mechanisms that can be guaranteed for operating systems, configuration data and applications on user devices [7], it is necessary for the network infrastructure to provide some level of protection against these types of assaults.

Mainly there are various kinds of DoS and DDoS attack scenarios that can be executed on a targeted device. The classification of these attacks can be done in number of ways. One of the primary classification that is available is shown in figure .DoS/DDoS attacks can fall into one of three categories: volume-based, protocol, or application layer [41]. The volume-based DoS

attacks comprise of attacks that deplete the target's bandwidth through the delivery of massive volumes of traffic. A typical examples for these type of attacks are ICMP flood, UDP flood etc. Protocol attacks are the type of attacks that take use of the weaknesses that are present in the protocols. Two common types of protocol attacks are the SYN flood and the Smurf attack. Application layer attacks are so-called because they concentrate on exploiting application layer services and protocols. Prominent examples for these kinds of assaults include HTTP flood attacks and slow rate DoS attacks [41].

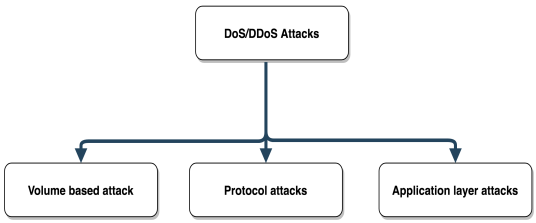


Figure 2.6: DoS/DDoS Classification

To ensure that all types are covered, attacks such as ICMP flood, UDP flood, SYN flood, HTTP flood, and slow rate DoS were chosen to be carried out for data collection in our dataset. A description on each of the attack scenarios is explained below,

2.3.2.1 ICMP flood

An ICMP flood attack typically involves ICMP echo requests that are sent to a target at a very high rate. In a normal network ICMP echo requests are used to ping a device to diagnose the health and connectivity between two devices. An ICMP reply is often received when such a request is sent. For this response to be sent, the destination server may need to allocate some resources to process each request. In ICMP flooding large number of ICMP echo requests are directed toward the target at a very high frequency. As a result, the targeted server has to use its resources in order to respond to the ICMP echo requests. This results in massive amounts of traffic that would overwhelm the network.

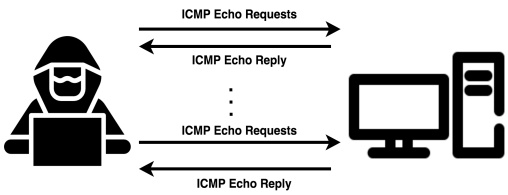


Figure 2.7: ICMP flood

2.3.2.2 UDP flood

This attack is quite similar to the ICMP flood, but instead of using ICMP packets, it uses UDP packets to overwhelm the target server. In contrast to TCP, the UDP protocol is a connection-less and session-less protocol. This means that no initial connection needs to be established before communication between two devices. Since enormous amounts of "best effort" traffic can be sent across UDP channels to any site without any limitation on the rate, UDP might be considered to be a protocol that is more susceptible to vulnerabilities. Because of this, UDP flooding is extremely effective regardless of the network configuration being used because it can be carried out with a minimal amount of resources. In a UDP flooding scenario, once the UDP datagram is received at the target, it checks for a associated application. If there exists no such application, a 'Destination Unreachable' packet will be sent back to the attacker. As large amounts of UDP datagrams are received and responded, the system may become unresponsive and unavailable for other users with time.

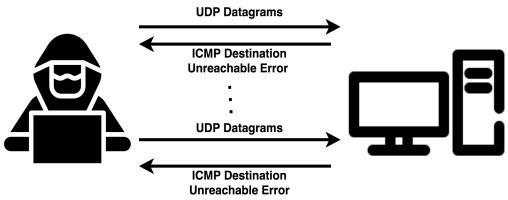


Figure 2.8: UDP flood

2.3.2.3 SYN flood

The TCP three way handshake is exploited in these type of attacks as was the case in SYN scan and TCP connect scan. In a TCP handshake, initial connection is made through a SYN packet sent to the destination. In response to this, a SYN ACK is received from the destination. In order to complete the three way handshake another ACK is sent as a response to the SYN ACK received. This establishes a TCP connection between two devices. In TCP flood, the initial two steps are completed. However, the final step of establishing the connection is not performed by the attacker. This leaves the port half open while large number of SYN packets are been continuously sent to the target. The arrival of each new SYN packet forces a new half open connection for a certain length of time with the attacker. Once all accessible ports are occupied, the server will be incapable of responding to any legitimate user.

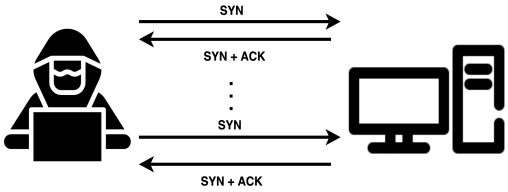


Figure 2.9: SYN flood

2.3.2.4 HTTP flood

A HTTP flood attack is a popular DoS/DDoS attack method which is used to attack web servers and applications. Hence it is considered to be application layer attack according to the classification in figure 2.6. An application layer attack uses vulnerabilities in the application layer protocols for exploitation. Among application layer protocols, HTTP exists as one of the most widely spread protocols due to its capability of integration with online services [41]. HTTP flood is one of the most effective forms of DoS attacks due to the wide use of the protocol and security equipment not blocking them by default. The HTTP flood attacks can be effective to mimic human behavior so that attacks may go undetected.

The HTTP flood can take place in two different methods by sending HTTP-GET requests or HTTP-POST requests. GET requests, which are used in HTTP, are used to obtain static content like images, whereas POST requests are used to access dynamically produced resources like forms. HTTP flood attacks do not utilize packet spoofing or packet malformation as some other attacks. However, it needs less bandwidth compared to other methods for bringing down a target.

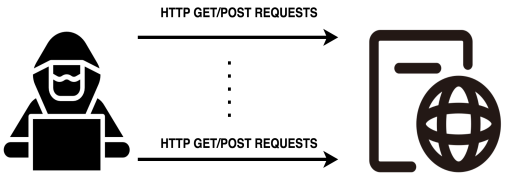


Figure 2.10: HTTP flood

2.3.3 Low rate DoS/DDoS

The low rate DoS attacks differentiate from other attacks due to its low speed compared to the comparatively high flooding rates seen in other attacks. These slow and low attacks targets on the application layer through the use of HTTP protocol in most cases. The attack types in LDoS can be categorized into two as slowloris and slow POST attacks. Slowloris attacks are carried out by establishing several connections with a targeted web server and maintaining them for as long as possible. This is accomplished by sending HTTP partial headers continually without

completion [42]. The web server will keep opening large number of connections while waiting for each request to complete, which never occurs. This continues until the server encounters a communication timeout or the attacker terminates the attack. Eventually, server resources are exhausted to the point where genuine connections cannot be serviced.

An attacker will send genuine POST requests to a web server in slow POST attacks. In these requests, packet size will be specified as a large value in the packet header. However, the message's body will be sent at a very slow rate sometimes as low as one byte per each transmission [42]. The server will wait for the complete duration of the message that is indicated in the header. This will render server resources inaccessible until the request has been fulfilled.

LDoS attacks are difficult to detect due to their slow and low nature, which closely resembles normal traffic. In addition, the sending of partial requests rather than malformed packets makes it easier to evade conventional intrusion detection techniques.

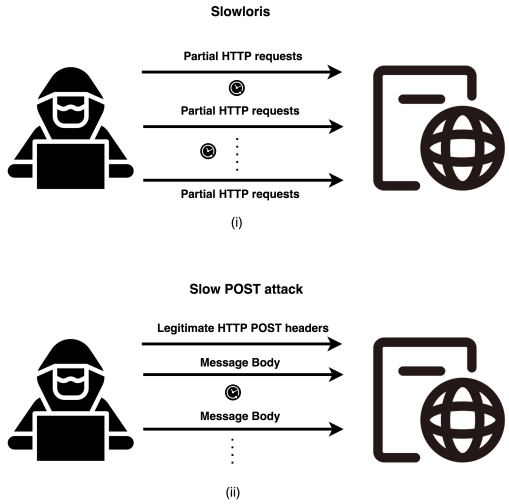


Figure 2.11: Slow Rate DoS attacks (i) Slowloris (ii) Slow POST attack

2.4 Tools

During this work, different tools were used at each stage of the data collection process. Initial data collection was carried out in packet based format and hence *Wireshark* tool was used to open and examine packets. Following this, the packets had to be processed in order remove GTP protocol from them using *Tracewrangler* tool. *Argus* tool was used extensively in order to create network flows from the packets and then to convert those flows into a csv.

The different types of attack scenarios were executed through a variety of tools. Port scans were carried out through the Nmap open source tool. Hping3 was used to execute flooding attacks such as ICMP flood, UDP flood and SYN flood. LOIC, Goldeneye, HULK, Slowloris and Torshammer was used to conduct application layer attacks.

2.4.1 Wireshark

Wireshark is a well known network packet analyzer that is capable of performing multitude of functions. Wireshark is most often used for packet capture in a network of even multiple devices. The packet sniffing of multiple devices mainly depends on the location of the host with Wireshark in the network. In addition to packet capture, Wireshark can also filter packets based on several properties. Through the use of filters, only the necessary data can be examined and even extracted into a separate file. The ability of Wireshark to portray packet information in a graphical format that can be comprehended by any user is also a significant advantage that it provides.

Wireshark can be considered an essential tool for network performance analysis. It can be used to troubleshoot networks with performance issues and assist security professionals in monitoring suspicious network transactions.

Wireshark uses the standard file formats:pcap and pcapng as the default formats for read and write. In addition to that, it has the ability to read and write to a number of other formats supported by other capturing tools.

2.4.2 Tracewrangler

Tracewrangler is a Windows-based network capture toolkit that supports the same file formats:pcap and pcapng. This tool's primary purpose is the sanitization, anonymization, and scrubbing of packets created by Wireshark/tcpdump, etc. It can also be used to editing packets in large quantities at once by removing layers such as GTP, MPLS, GRE etc. The tool has the ability to merge and aggregate pcaps recorded in more than one interface.

2.4.3 Argus

Argus is an open source project which aims at delivering network flows from packet based formats. Since its inception in the late 1980s at Carnegie Mellon's Software Engineering Institute by Carter Bullard, Argus has been an active and essential component in the development of network flow technologies for modern networking and cyber security. The Argus tool seeks to handle a variety of network flow data processing concerns, including scale, performance, applicability, confidentiality, and utility.

Argus is a system for network auditing which comprises of two packages. A packet processing network flow sensor argus, which generates Argus flows and a collection of argus data processing programs called argus clients which can be used together in order to build high quality data flow channels that could process network data in real time or even uncoupled to assist in large-scale data analysis. Argus clients supports a set of different functions and operations on streaming network flows as well as files. Some of the basic client programs include, ra, racluster, rasplit, rasort, raconvert which are used to print, cluster, split, sort and convert to other file formats respectively.

2.4.4 Nmap

Nmap is one of the most popular network scanning tools used for penetration testing, port scanning and network mapping. Nmap uses raw IP packets to determine the status of ports in

a host through multiple different scanning techniques. Nmap can be run on all major operating systems including Windows, Linux and Mac OS X and also supports some older operating systems such as Solaris, AIX and AmigaOS.

Various types of port scanning can be performed through Nmap including SYN scan, TCP connect scan and UDP scan. The user/attacker can specify the command to execute the type of attack through the command prompt. The IP address of the target and the type of attack will need to be specified in these commands. Typically, it is possible to scan all the ports of a given IP address or specify a range of ports to be scanned. Scanning all ports may take a considerable amount of time depending on the type of port scan. Furthermore, users can set the depth of the scan to a light scan or more detailed one.

2.4.5 Hping3

Hping3 is another network tool that can transmit customized ICMP/TCP/UDP packets which could be used to perform a DoS attack. Hping3 is able to tolerate fragmentation as well as arbitrary packet payload sizes, which may enable DoS attacks to circumvent the network security systems. The tool has the capability to perform DoS attacks such as ICMP flood, UDP flood and SYN flood with varied number of packet sizes and fragmentation.

The tool may also be used to test the network performance using various protocols and packet sizes. In addition it can also be used to perform port scanning, remote OS fingerprint, testing firewall rules, path MTU discovery etc. The commands to perform different attacks are executed through a command line interface.

2.4.6 LOIC

Low Orbit Ion Canon, abbreviated as LOIC, is a well-known open source tool utilized for network stress testing and DoS/DDoS assaults. An attacker can use this tool to carry out DoS attacks by flooding the target with TCP, UDP or HTTP GET requests. With its graphical user interface, an attacker only needs to enter the target IP address and the intended port.

Once launched, LOIC generates massive number of traffic and sends them continuously to the targeted server. As the quantity of message requests increases, the server becomes overloaded and is eventually unable to serve normal users.

2.4.7 Goldeneye

Goldeneye is another python based tool used to perform DoS attacks involving HTTP GET requests. Goldeneye is a application layer attack tool which attempts to keep the connections alive along with cache control options to persist the socket connection smashing via caching until all available sockets have been consumed.

The tool is powerful for carrying out DoS attacks as a single machine has the capability to take down a server. After making a full TCP connection, Goldeneye needs only a few hundred legitimate HTTP requests to disrupt a web server.

As the tool is based on python, the python file should be called from the command line along with the target IP address in order to perform this attack.

2.4.8 HULK

HULK is a DoS tool designed to attack web servers by producing unique and masked traffic volumes. The word HULK stands for "HTTP Unbearable Load King" and also bases its attack on HTTP GET requests. HULK flood differentiates from the majority of other DoS/DDoS attack programs, which generate easily detectable, repeatable patterns. The HULK flood operates on the principle that a unique pattern is generated for each request, with the purpose of raising server load and eluding intrusion detection and prevention systems.

This attack is also conducted using HTTP requests that attempt to maintain open connections by utilizing Keep-Alive with a flexible time window. When the maximum number of simultaneous connections allowed by a server is reached, the server is unable to fulfill the requests of any further users.

2.4.9 Slowloris

Slowloris is another DoS attack program that is used to attack vulnerabilities present in the application layer through the use of partial HTTP requests. This attack type falls under the low rate DoS attack type. The attack operates by establishing and maintaining connections with a targeted Web server for as long as possible. Slowloris is a specific attack tool designed to allow a single computer to bring down a server without consuming a massive amount of bandwidth. Slowloris utilizes a negligible amount of bandwidth and attempts to drain server resources by delivering requests that seem slower than usual but are otherwise nearly identical to authentic user traffic.

On the server, there will be a limited number of threads available to manage concurrent connections. Each server thread will attempt to maintain its existence while awaiting the conclusion of the slow request, which will never occur. Once the server reaches its maximum number of connections, future connections will not be accepted, resulting in a denial of service.

As slowloris is also available as a Python script, its execution entails invoking the Python file along with the IP address of the destination from the terminal.

2.4.10 Torshammer

Torshammer is also a slow rate DoS attack tool that attacks a web server using slow POST requests. It can be described as an efficient and disruptive attack tool. In these slow POST attacks, the attacker will issue POST requests very slowly in order to direct the server into keeping the connections open. As POST requests are commonly used to submit data to a server, the server will typically allocate a longer connection timeout [43]. An attacker will take advantage of this to send repeated POST requests at very slow rates, thereby establishing new connections.

Torshammer is also python based and can be executed through the terminal with a simple command together with the IP address of the target server.

3 IMPLEMENTATION

This chapter will discuss the data collection stage of the work. The 5G Test Network at University of Oulu was used as the core network in this instance. The architecture of the network will be discussed in more detail during this chapter. Further, the generation of benign traffic and the attack scenarios are discussed extensively. All commands used from different attack tools are also described individually.

3.1 Network Architecture

The data was collected on a real 5G network with a variety of capabilities. The network architecture consisted of mobile users, 5G modems, PICO base stations, Raspberry Pi's, linux servers, firewalls and switches. In order for the network design to accurately mimic a real-world scenario, it was necessary to isolate the attack network and the target network. This is because the attack threat in modern networks with an unimaginable number of devices could originate from outside of the network in most cases. In addition, the attacks could originate from a variety of entry points and be directed toward the same target. To illustrate this, the attacks were carried out by two attackers on two separate networks. This further demonstrates the necessity of a coordinated global security mechanism for future wireless networks.

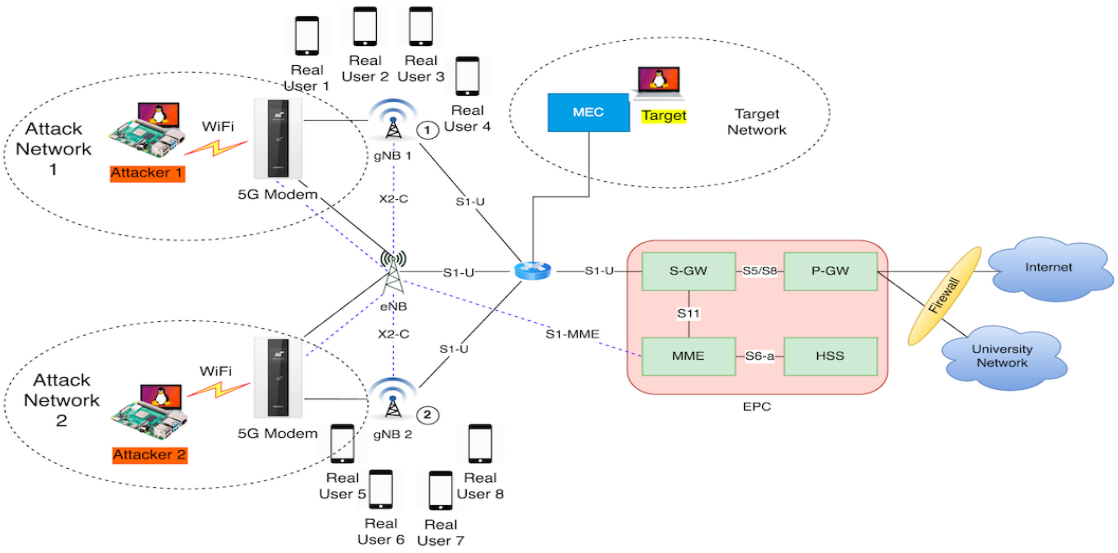


Figure 3.1: Network architecture of the testbed

In the attacker network, a Raspberry Pi 4 Model B computer running Ubuntu operating system was used to execute the varied attack types. This attacker was connected to the Wifi through a modem which is connected to the 5G pico base stations. The use of Raspberry Pi computer enabled us to carry out different types of attacks effectively from a separate network.

The victim/target network was deployed in the MEC. MEC is an integral part of the 5G networks since it pushes application hosting from centralized data centers to the network's edge. This is considered vital for 5G networks to satisfy its requirements, which include low latency and bandwidth to support massive numbers of IoT devices and mission critical applications

[44]. Having the target PC placed at the MEC permits us to establish a distinct network for the victim and to replicate a real-world 5G network scenario that employs additional supporting technologies such as the MEC. A PC running Ubuntu was placed at the MEC to act as the target during the data collection process.

In this network, two Pico base stations are present to which the two attack networks are connected. A pico base station, also known as a pico cell, is a compact cellular base station (BS) that serves as an alternative to a repeater predominately in indoors. It is often used to extend wireless coverage to areas that cannot be accessed by networks served through large cell towers, such as the inside of buildings or remote locations.

The two pico base stations are connected to the eNodeB which is the main macro base station at University of Oulu through the x2-c interface. Typically, the x2-c interface between eNBs is responsible for functions such as mobility management, changeover preparation, status transfer, UE context release, handover cancellation, inter cell interference cooperation, and load management. The pico base stations are also connected to the switch through the S1-U interface.

Table 3.1: Network devices

Network Device	Description
Attacker	Raspberry Pi 4 Model B with Ubuntu 21.10 - Impish
Victim	HP with Ubuntu 18.04.5 LTS - Bionic
Base Station	Nokia - Flexi Zone Indoor Pico BTS
Modem	HUAWEI - E6878
Mobile devices	Motorola - Moto g50 5G
Switch	Dell - N1524

3.2 5G Test Network Finland (5GTNF)

5G Test Network Finland is a developing, open innovation ecosystem that supports technological development beyond 5G [45]. The overall objective of the 5G Test Network Finland (5GTNF) is to bridge the gap between laboratory-based 5G and beyond testing environments and commercial network deployments. Additionally, the 5G Test Network Finland (5GTNF) will provide trialing support as well as tailored infrastructure configurations for the telecommunications industry, vertical industries, and the scientific community. The primary focus of the 5GTNF is to incorporate inter-disciplinary competences, such as beyond 5G network and radio enablers, cyber security, use of AI and develop technologies related to them. The 5GTNF is a collaborative effort of the industry, academics and the government of Finland [45].

In this work the 5GTNF site at University of Oulu was used as the underlying infrastructure in the 5G testbed. The Non Stand Alone architecture for 4G and 5G dual connectivity specified in option 3a of [46] is used in this network. In non-standalone (NSA) architecture, NR radio cells and LTE radio cells interoperate via dual connectivity to provide UEs with combined radio access. In this architecture, both the eNB and the gNB can connect directly with the EPC.

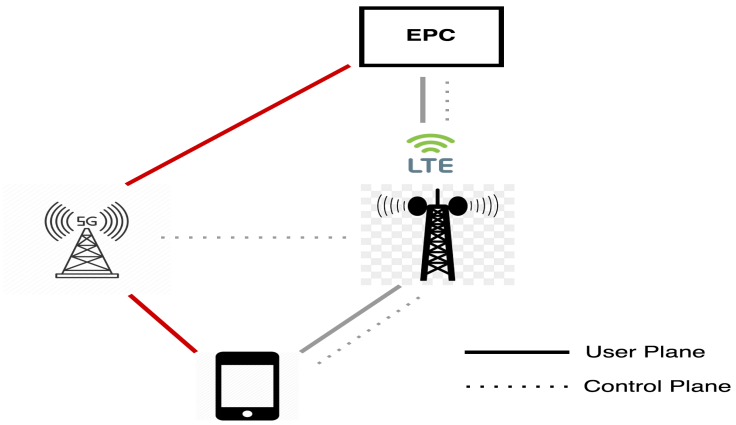


Figure 3.2: 5G Implementation using option 3a

3.3 Benign Traffic

This dataset is distinguished by the availability of benign traffic from genuine devices as opposed to simulations as seen in majority of previous work. In several of the past studies, the benign traffic from devices have been captured at a prior time and the traffic profiles have been later introduced in the network during the period of the attacks. This may have a impact on the dataset in terms of the source IP address, timestamps and other crucial characteristics expected in a live network. In contrast, the benign traffic within this network is generated in real-time from the devices to represent a more realistic scenario.

In addition, this is the first time that mobile devices are deployed to build normal network traffic in major datasets that are available. The mobile devices were connected to each of the pico base stations in order to generate the traffic. The *Network Monitor* tool was used to make sure that each mobile device is connected to the relevant pico base station.

The mobile devices were used to provide traffic in different types of protocols such as HTTP, HTTPS, SSH and SFTP to account for variety. The HTTP and HTTPS traffic was generated through live streaming services as well as web browsing. SSH and SFTP was deployed through SSH clients and servers installed on mobile devices. It was ensured that the normal traffic is available at all times during the data captures and attacks.

3.4 Attack Scenarios

The attacks were operated from the Raspberry Pi 4 Model B computers with Ubuntu 21.10 running. Attack tools such as Nmap, Hping3, LOIC were installed on the Raspberry Pi in their newest versions. The python scripts of Goldeneye, HULK, Slowloris and torshammer were called with the required python versions.

The port scans were executed from the Nmap tool. The tool is available for download from the internet as an open source tool. The SYN scan, TCP connect scan and UDP scan were executed from the terminal in Ubuntu. The target was set to the destination IP address of the victim deployed at the MEC. In all the instances, the IP address of the target was *10.41.150.68* The network architecture was same as indicated in the Figure 3.1. The following commands

were given to execute port scans from Nmap tool.

SYN Scan

sudo nmap -sS 'Target IP address' -p 'Targeted port / range of ports'

TCP Connect Scan

sudo nmap -sT 'Target IP address' -p 'Targeted port / range of ports'

UDP Scan

sudo nmap -sU 'Target IP address' -p 'Targeted port / range of ports'

The *-sS, -sT and -sU* in these commands stands out for the different types of attacks. The *-p* is used to specify the targeted ports. If no value is provided for this, the tool will carry out the scan for all the ports in the target.

The Hping3 tool was used to carry out different types of volume based DoS attacks as well as protocol based DoS attacks. Specifically, they were ICMP flood, UDP flood and SYN flood. In all these attacks, the target is flooded with different types of packets at a very high rate. Hping3 is also a freely available tool and commands were executed from the Ubuntu terminal.

ICMP flood

sudo hping3 –flood –rand-source -1 -p 'Targeted port / range of ports' 'Target IP address

UDP flood

sudo hping3 –flood –rand-source –udp -p 'Targeted port / range of ports' 'Target IP address

SYN flood

sudo hping3 -S -p 'Targeted port / range of ports' –flood –rand-source 'Target IP address'

Here, the *–flood* command provides frequency of sending packets while *–rand-source* directs the tool to send packets with hidden source IP or more specifically with random IP addresses. The *-1. –udp, -S* suggests the type of protocol to be used when flooding with packets. The targeted port or range of port is specified after -p. In addition to this, a user can set the values like packet size, packet count, interval between packets etc.

The HTTP flood attacks were performed using multiple tools. This include the LOIC, Goldeneye and the HULK tool. Before carrying out these attacks, a web server had to be installed at the target. The web server that was used for this purpose was Apache2 web server. Apache is the most regularly used web server for linux systems. Apache2 web server was installed on the Ubuntu machine which acted as the target in the previous scenarios. After the installation, the firewall was configured in order to let outside parties access the web server.

The HTTP flood attacks were then carried out with the target set to the IP address of the web server. Among other tools, the LOIC tool comes with a graphical user interface where the target IP and the target port can be specified as shown in figure 3.3. Here, target IP address was 10.41.150.68 and the target port was set to port 80.

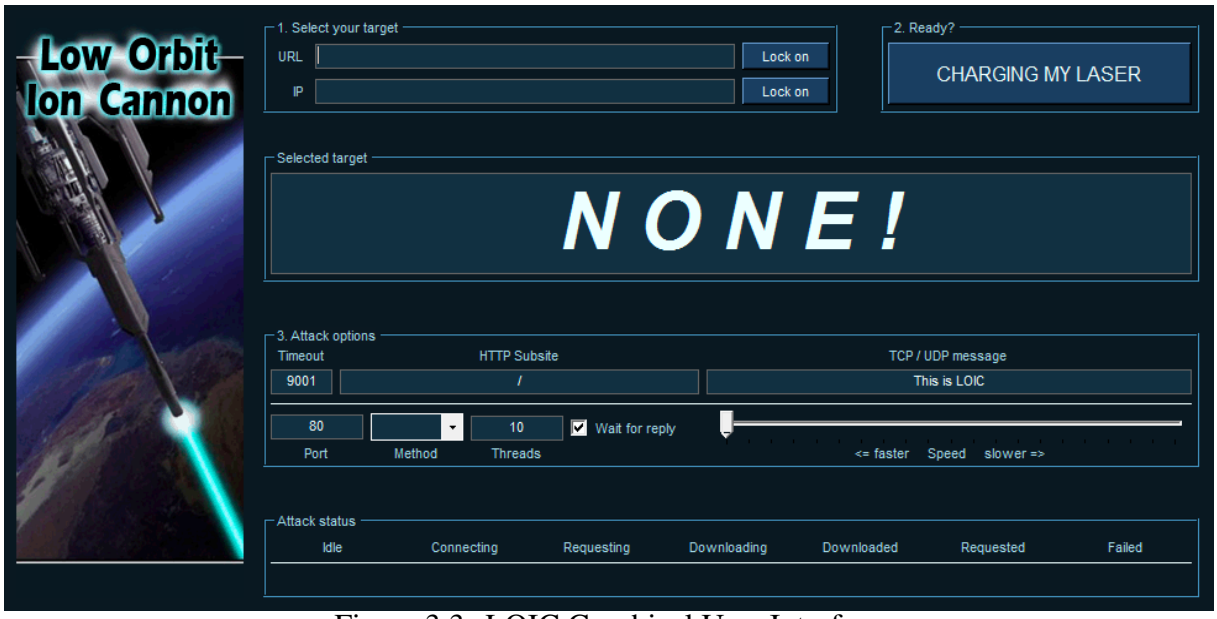


Figure 3.3: LOIC Graphical User Interface

Goldeneye is another tool used to carry out the HTTP flood attacks. It is a script written in python and therefore the file needs to be called from the terminal along with other speifications. The following command was given from the Ubuntu terminal.

Goldeneye

sudo python3 ./goldeneye.py http://'Target IP address of the web server'

The target IP address of the Apache2 web server was set as *http://10.41.150.68* in this instance.

HULK also acts in a similar manner as it is also a python script that needs to be called through the terminal. The following command was used in order to generate an HTTP flood using HULK.

HULK

sudo python3 hulk.py http://'Target IP address of the web server : Targeted port'

Similar to the Goldeneye, the IP address of the web server was given in this one. The targeted port was set as port 80.

The slow rate DoS attacks were carried out by using Slowloris and Torshammer which were also scripts based on python. For the slowloris attack, the following command was given through the Ubuntu terminal after getting into the directory of the python file.
The target IP address was specified to be *10.41.150.68*

Slowloris

python3 slowloris.py 'Target IP address of the web server'

Similarly, the slow rate DoS using HTTP POST requests was executed through the Torshammer python script. This was based on python version 2 and had to be called using the following command.

Torshammer

python2 torshammer.py -t 'Target IP address of the web server'

3.5 Data collection

The data collection was carried out on two consecutive days. For the purpose of data collection, a PC was connected to the base station through its management port.

Initially, different types of DoS/DDoS attacks were carried out. It was made sure that the data capture was done for 30 minutes from both the base stations for each type of attack. Throughout this time, the benign traffic was available continuously. During this 30 minute period for data collection, an attack was carried out for 10 minutes from attackers connected to each of the base stations. This was done such that there exist a time period where the attacks are carried out from both base stations simultaneously as well as other time periods where only an attacker connected to one individual base station is carrying out the attacks. This scenario is understandable from the table 3.2.

Table 3.2: Schedule of the DoS/DDoS attacks

Type of DoS/DDoS Attack	Date	Period of data collection	Period of attack (Base Station 1)	Period of attack (Base Station 2)
Slow rate DoS - Slowloris	16/6/2022	2.45pm-3.15pm	2.55pm-3.05pm	2.50pm-3.00pm
HTTP flood - LOIC	16/6/2022	3.30pm-4.00pm	3.40pm-3.50pm	3.45pm-3.55pm
SYN flood - Hping3	16/6/2022	4.30pm-5.00pm	4.35pm-4.45pm	4.40pm-4.50pm
HTTP flood - Goldeneye	16/6/2022	5.30pm-6.00pm	5.35pm-5.45pm	5.50pm-6.00pm
Slow rate DoS - Torshammer	17/7/2022	10.30am-11.00am	10.40am-10.50am	10.45am-10.55am
HTTP flood - HULK	17/6/2022	11.30am-12.00am	11.40am-11.50am	11.45-11.55
UDP flood - Hping3	17/6/2022	12.15pm-12.45pm	12.25pm-12.35	12.30pm-12.40pm
ICMP flood - Hping3	17/6/2022	2.00pm-2.30pm	2.10pm-2.20pm	2.15pm-2.25pm

After, the collection of data from DoS attacks, the data was collected for port scans. Here, each data collection cycle was carried out for 10 minutes time. Here also, we ensured that benign traffic is deployed at all times from the mobile users. During this time period of 10 minutes, multiple number of port scans were executed as the time taken for a port scan was quite short. The table 3.3 shows the schedule of the port scans that were carried out.

Table 3.3: Schedule of the Port Scans

Type of Port Scan	Date	Period of attack (Base Station 1)	Period of attack (Base Station 2)
SYN Scan	17/6/2022	3.00pm-3.10pm	3.00pm-3.10pm
TCP Connect Scan	17/6/2022	3.20pm-3.30pm	3.20pm-3.30pm
UDP Scan	17/6/2022	3.45pm-3.55pm	3.45pm-3.55pm

4 DATA PROCESSING

The data collected from the base stations were available as pcap files for each of the attacks separately. Therefore, 22 pcap files were collected in total from both base stations with 11 from each of the base stations. The overall time in these data accounted for nearly 9 hours.

Data pre-processing can be considered an important step in achieving the best results in machine learning models. The raw data collected should be converted to information that can be fed into a machine learning model. This can involve several steps. The figure 4.1 shows the main steps that were involved with our work.

Figure 4.1: Steps for Data Processing before feeding onto the ML models

4.1 GTP layer removal

The data collected from the two base stations were first analysed through the Wireshark tool. A portion of the acquired data that was available in packet-based format contained the GPRS Tunneling Protocol (GTP) layer. GTP is a form of protocol utilized by IP-based communication protocols in order to transport General Packet Radio Service (GPRS) across GSM, UMTS, LTE, and 5G New Radio (NR) networks. Again, GTP can be divided into three distinct protocols: GTP-C, GTP-U, and GTP' (GTP Prime). GTP-C is used for communication between gateway GPRS support nodes and serving GPRS support nodes within the GPRS core network. On the other hand, GTP-U is used to transport user data inside the GPRS core and between the radio access network and the core network. GTP' (GTP prime) employs similar message structure to GTP-C and GTP-U, but has a different purpose. It can be used to deliver charging data from the GSM or UMTS network's charging data function (CDF) to the charging gateway function (CGF).

The data collected from the base stations contained packets with GTP-U protocol. This needed to be removed in order to reflect the actual protocols in the network flows which were generated subsequently. Therefore, in order to remove this GTP layer, the TraceWrangler tool was used which had the capability to remove the GTP layer in large batches of packets. The packets without the GTP layer was saved again in the pcapng format for each individual attack.

4.2 Conversion to network flows

After the removal of GTP layer, the data was available in the pcapng format which is a packet based format supported by Wireshark. Training of data in either a packet-based format or a flow-based format can be utilized for intrusion detection. Both strategies have been extensively practiced in earlier studies, demonstrating that both can have favorable effects in varying aspects.

Deep packet inspection or packet-based intrusion detection will discover intrusions based on a combination of packet header and payload analysis and scans [47]. Typically, this strategy is implemented using tools like tcp dump. Signature based intrusion detection most often use this strategy to spot malicious packets. Despite the fact that large amounts of data are analyzed and processed, the drawback of this type of a approach is that it takes a considerable amount of time to scan headers and payloads of every single packet notably in the context of today's high speed networks. Therefore, to prevent packet-based intrusion detection schemes from becoming a network bottleneck, a high processing throughput is required [48]. However, sophisticated systems that can monitor every single packet in a high-speed network may be extremely costly and resource-intensive [47].

A flow is considered as a unidirectional stream of packets between two devices with identical source and destination IP addresses, source and destination ports, and protocol [47]. In addition to these characteristics, different tools may incorporate different features for the purpose of classification of flows. Flow based detection approaches are been commonly deployed for machine learning based intrusion detection. In contrast to packet based formats, the flow based formats provides a high-level description about the communications in the network while protecting the privacy of the users [49]. The approach is also regarded to be more efficient since the amount of data and size of data is drastically reduced compared to packet-based formats. This will significantly reduce the processing capabilities required for intrusion detection. Consequently, performance difficulties associated with flow-based approaches are not a concern in high-speed networks. Furthermore, the growing numbers of threat surfaces has resulted in attacks that cannot be identified through a single packet but spans across a multiple number of packets. These attacks and other zero day attacks can effectively be identified through flow based approach rather than a packet based one.

In this work, the packet based data collected was converted into a flow based format without loosing the most important features. This was done using the Argus tool. Argus supports generation of network flows in real-time as well as conversion of pcap files into network flows. The following commands were used in order to convert the packets into flows. Each of the files available were converted separately.

Conversion of packets to flows

/usr/local/sbin/argus -r filename.pcap -w filename.argus

Read and display the network flows in a Argus file

/usr/local/bin/ra -r filename.argus

The *-r* specifies the name of the file to be read and the *-w* specifies the file that data is been written. The network flows available in the argus file format was then written into a csv file. All the fields, that are extracted through the argus tool were written into the csv file using the following command.

Writing into a csv file

/usr/local/bin/ra -c , -s srcid rank stime ltime trans flgs seq dur runtime idle mean stddev sum min max smac dmac soui doui saddr daddr proto sport dport stos dtos sdsb ddsb sco dco sttl dttl shops dhops sipid dipid smpls dmpls autoid sas das ias cause nstroke snstroke dnstroke pkts spkts dpkts bytes sbytes dbytes appbytes sappbytes dappbytes pcr offset smeansz dmeansz load sload dload loss sloss dloss ploss psloss pdloss retrans sretrans dretrans pretrans psretrans pdretrans sgap dgap rate srate drate dir sintpkt sintdist sintpktact sintdistact sintpktidl sintdistidl dintpkt dintdist dintpktact dintdistact dintpktidl dintdistidl sjit sjitact sjitidle djit djitact djitidle state label suser duser swin dwin svlan dvlan svid dvid svpri dvpri srng erng stcpb dtcpb tcprtt synack ackdat -r filename.argus > filename.csv

In here the *-c* specifies the type of separator to be used when writing into the csv file and *-s* specifies the fields to be written. In total, 116 features were extracted and written into the csv format. This includes some main features such as source and destination IP addresses, source and destination port numbers, protocol, number of packets, packet size in bytes etc. [1]

A total of 22 csv files were available for all attacks in both the base stations with each having 11. All the aforementioned fields were present in each and every csv file.

4.3 Data aggregation and labelling

The separately available data for each attack had to be combined in order for further analysis. The data was first aggregated based on the base station from which the data was collected as this will be beneficial for implementation and testing of federated learning based security solutions. Then the data from each base station was combined to generate the total number of network flows. Before the aggregation, the data was labeled based on 2 characteristics for further analysis which included a binary classification as well as a multi class classification.

First, the labelling process considered whether a flow was malicious or benign. This was decided based on the source IP address and the destination IP address. As the source IP address and the destination IP address were already known, the condition was set to decide a flow is malicious if either of the source IP or destination IP matches with the known IP addresses of the victim or attacker and then the remaining one of the source IP or destination IP matches with the one remaining out of victim and attacker. Flows from both directions were considered malicious since the responses from target to the attacker also resulted due to the attacks. In the dataset, the column 'Label' specified whether a flow is malicious or benign with the use of numbers 1 and 0 respectively.

Table 4.1: Meaning of values specified in the 'Label' column of the dataset

'Label' categories	Meaning
0	Benign
1	Malicious

[1] All features generated by Argus tool along with their meanings are available at `https://manpages.ubuntu.com/manpages/trusty/man1/ra.1.html`

The type of attack was specified using the same rules used for classification in the previous scenario with the separate csv files allowing in identification of the attack type. The identification of the attack type is beneficial in the multi class classification that will be performed. The 'Attack type' column in the dataset identified benign traffic, port scans, UDP flood, ICMP flood, SYN flood, HTTP flood and Slowrate DoS attacks and is represented as 0,1,2,3,4,5 and 6 respectively in the dataset. Furthermore, each type of the 11 attacks were also classified separately in another column for our own reference as well as the identification.

Table 4.2: Meaning of values specified in the 'Attack Type' column of the dataset

'Attack Types' categories	Meaning
0	Benign
1	Port scans
2	UDP flood
3	ICMP flood
4	SYN flood
5	HTTP flood
6	Slowrate DoS

Following the labelling process, data was aggregated by merging the separate csv files into one through a python script. Separate datasets were initially created for data collected from different base stations and later combined them to make one dataset. The data collected from either of the base stations are available separately as well for the purpose of federated learning based research.

The total number of flows in the dataset was 851205. Among these flows the following attack variety was seen.

Table 4.3: Composition of the Dataset

Type of Attack	Number of flows
Benign traffic	32995
Port Scans	32313
UDP flood	223401
ICMP flood	266816
SYN flood	155986
HTTP flood	96249
Slowrate DoS	43444

4.4 Encoding

The combined dataset consisted of all the fields in the table **??** along with the three additional columns used for the purpose of labelling. The combined dataset contained 851205 records of network flows in total. Most of the fields in the dataset contained numerical figures while several categorical data also existed.

In machine learning, categorical data is frequently employed for classification and regression problems. However, the only form of data that can be fed onto a machine learning model as an input is numerical data [50]. Therefore, this demands encoding of categorical data into numerical values such that each categorical attribute is represented through a number [50].

Among different techniques available for encoding categorical data, we used one hot encoding in our dataset. One hot encoding is one of the most commonly used techniques in data encoding for machine learning as well as neural networks. One hot encoding works by transforming a particular variable of n observances and d distinct categories into d binary variables each with n observations [50]. Each of the d variable or category will contain either 1 or 0 representing the availability or non-availability of that category. The one hot encoding will increase the number of columns or fields in the dataset depending on the number of possible categories. Therefore, this may not be feasible when there is a large number of distinct categories.

In this dataset, as there were no variables with far too many categories in the fields that were considered, the one hot encoding technique was utilized to transform categorical variables to numerical variables. One more rationale for using one hot encoding was that it achieved acceptable accuracy levels with less complexity, as demonstrated in [50]. The variables 'Flgs', 'State' and 'Proto' were transformed into numerical variables before feeding onto ML models. These variables included 10, 16 and 3 distinct categories in each of them respectively. Therefore, a total of 29 extra fields were added onto the dataset as a result of one hot encoding. The table 4.4 shows the extra fields that were added into the dataset.

Table 4.4: Fields added from One Hot Encoding (Notations from Argus flow features given in [51] are are adopted

Variable	Categories
'Flgs'	*' e'*
	'e g'
	'e i'
	'e r'
	'e &'
	*'e *'*
	'e d'
	' e s'
	'e'
	'e s'
'State'	*CON*
	ECO
	FIN
	INT
	RST
	TST
	REQ
	RSP
	ACC
	ECR
	SRC
	PAR
	URP
	URH
	CLO

	NaN
'Proto'	*tcp* *udp* *icmp*

4.5 Feature selection

In most cases, a sizeable amount of data is required in order to train a machine learning model for the purpose of identifying and differentiating between legitimate traffic and malicious traffic. In the context of 5G, integrated technologies such as IoT and V2X will increase the frequency with which enormous amounts of data must be processed. The amount of data collected poses a significant challenge in resource management. The selection and storage of only the essential and adequate characteristics for AI/ML model training and tuning is one of the most straightforward methods for minimizing the volume of the datasets [52].

Therefore, feature selection is one of the most important aspects of data processing for any machine learning task, as it has a substantial impact on the accuracy of the resultant predictions and the training time. In order to predict the outcome of an unknown instance, machine learning models utilize a variety of features to learn. However, the possession of a large number of features, or high dimensionality in data, is mostly regarded as a challenge for the majority of machine learning models. In both theory and practice, feature selection has been shown to be effective in processing high-dimensional data and improving the learning efficiency [53]. It is regarded as an effective approach in saving computational time and improving learning accuracy through the removal of irrelevant and redundant data [54].

Feature selection mainly involves selecting a subset of features from an original set of features. This can be performed through different methods such as filter, wrapper and embedded methods. Filter methods are statistically based and operate independently of the characteristics and parameters of the model used for classification [55]. These statistical based methods mostly involve concepts such as correlation, mutual information and other statistical tests. The concept of filter method for feature selection is regarded as a general technique, and this generality could be interpreted as the potential applicability to any model [55]. In the wrapper technique, feature selection is carried out by observing the performance of the model under various feature configurations. In the majority of cases, prediction accuracy has been the primary selection criterion in these methods. The reliance on a specific classifier results in a loss of generality and bias, however adapting the set of inputs to local requirements typically improves the performance. Embedded approaches are based on algorithms inherent to the learning system for feature selection and elimination [55].

In this work, the filter methods were utilized owing to the generality it provides as different machine learning models will be applied for this dataset. The feature selection process was done in three steps. As the first step, some of the features were removed due to the complete unavailability of values whilst some features were removed due to presence of constant values. Most of the features that were removed due to the constant values were observed to be zeros. Moreover, in order to maintain the intrusion detection system's generality, certain features had to be eliminated. The intrusion detection should not rely on features such as the IP addresses of the target or attacker, destination and source port numbers, which in this instance were largely unique. In other words, the inclusion of these variables in the training dataset will make malicious traffic classification rather straightforward. Therefore, as the next step towards feature selection these features were removed in order to have a system capable of identifying and

detecting malicious traffic originating from any source and directing towards any destination. Finally, statistical techniques such as Pearson correlation coefficient and Chi-square test was employed in order to choose the final set of features.

4.5.1 Pearson Correlation Coefficient

The remaining features in the dataset comprised a number of features that provide similar information in determining the target variable. Therefore, in order to achieve the goal of feature reduction, one out of each such pair had to be eliminated. This task was achieved through the use of Pearson correlation. The Pearson correlation is a popular statistical method used to evaluate the linear correlation between two variables X and Y. The value of Pearson correlation coefficient exists in the range of -1 to +1 where a value of +1 suggests that X is completely positively linearly correlated to Y while a value of -1 implies complete negative correlation between the two variables as shown in the figures 4.2 [56]. If there exists no correlation between X and Y the correlation coefficient becomes 0. The following equation is used in order to calculate the Pearson correlation coefficient (PCC).

$$\rho_{x,y} = \frac{COV(X,Y)}{\sigma_x \sigma_y} = \frac{E(X-\mu_x)(Y-\mu_y)}{\sigma_x \sigma_y} \tag{4.1}$$

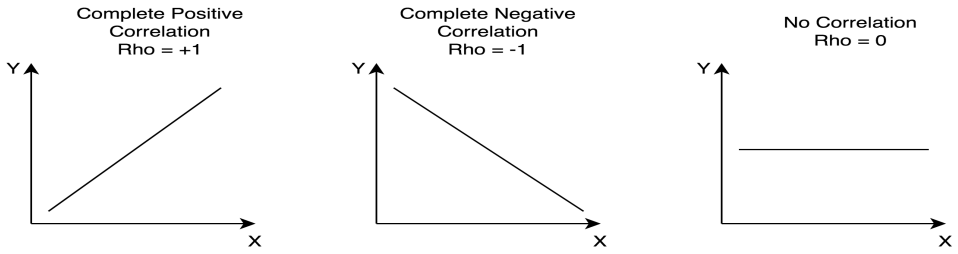


Figure 4.2: Pearson Correlation Coefficient for complete positive correlation, complete negative correlation and no correlation

In equation 4.1, the function $COV(X,Y)$ expresses the covariance between X and Y. μ_x and μ_y are the mean values of X and Y while σ_x and σ_y relates the standard deviation of X and Y. In this work, the Pearson correlation coefficient was obtained for every single pair of features in order to remove features with multicollinearity. This was carried out through a python script in Jupyter Notebook. The correlation was applied to the training set of the data. The figure 4.3 shows the heatmap that was generated along with the correlation coefficient values. The heatmap displays highly correlated values in darker colors, while less correlated values are depicted in lighter hues.

The observed data contained both negative and positive correlation values. In order to remove highly correlated data, the threshold was set to 0.90 with the correlation coefficients converted to absolute values in order to account for both positve and negative correlation. One feature

from each pair with a correlation greater than 0.90 was eliminated based on its correlation with the target. Out of a given pair of features, the eliminated feature was the one with the lowest correlation towards the target variable.

After the elimination of these redundant information, the Pearson correlation between the different variables and the target was also considered. Here, a higher correlation value means that the feature has a high impact on the output. The table 9.1 and 9.2 in Appendix 1 shows the PCC values obtained for the target variable in both binary and multi class classifications.

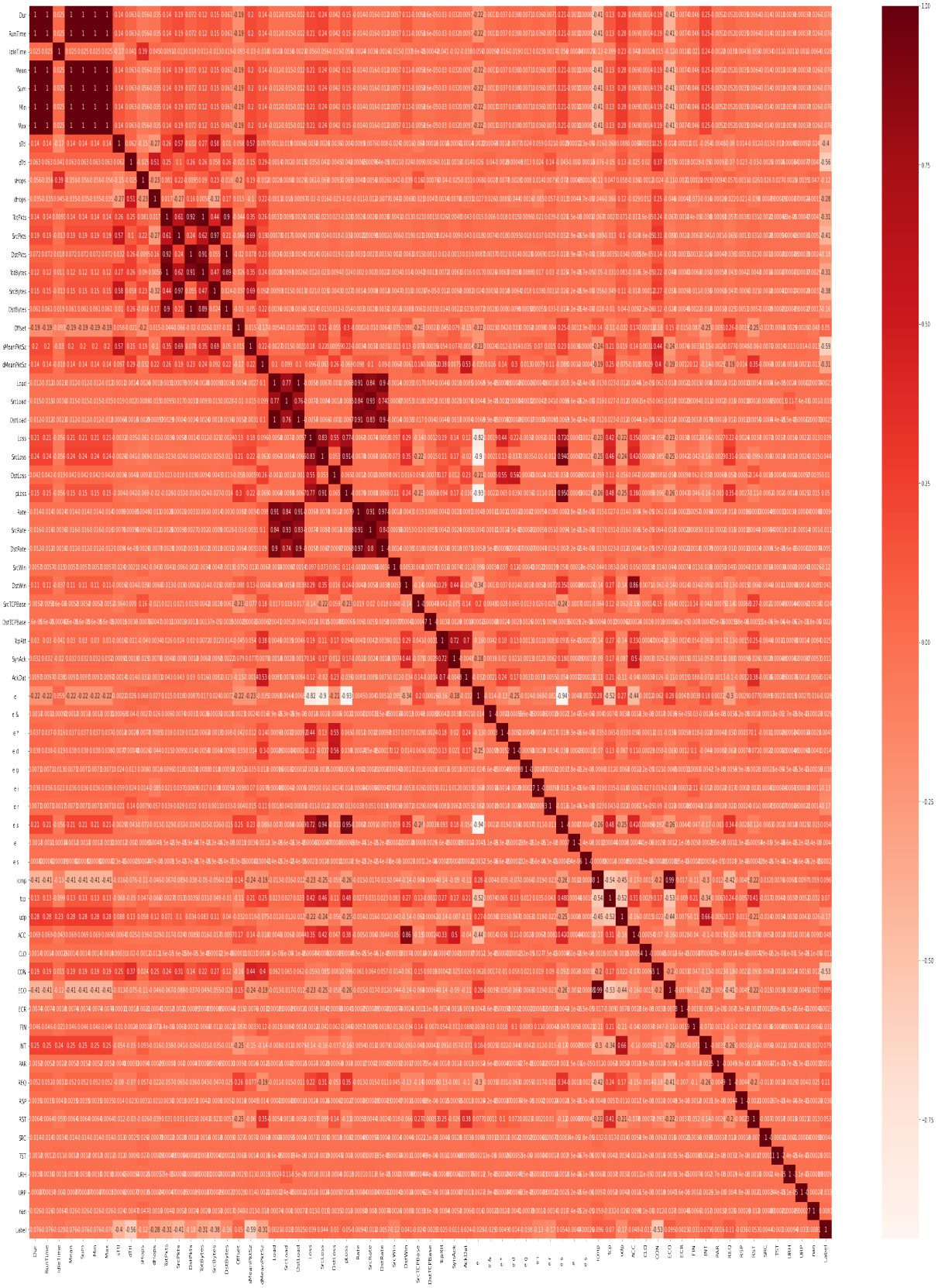


Figure 4.3: Heatmap with Pearson correlation between all features

4.5.2 Chi-square test

After the elimination of features using PCC, the best features in the dataset was chosen using a combination of PCC and Chi-square test. The Chi-square test is a another statistical test used mostly for the feature selection when categorical data are available.

In statistics, the chi-square test is used to assess the independence of two events. In equation 4.2, the O_i represents the observed values and E_i represents the expected values. In the case when two features are independent, the observed count will be closer to expected count which will result in lower chi-square values. A high chi-square value, on the other hand, suggests that the feature is highly reliant and should be considered for model training.

In this instance, the Chi-square test was used as the dataset contained some categorical features too. The tables 9.3 and 9.4 in Appendix 1 shows the ranking obtained from the Chi-square scores.

Although, the dataset contained categorical features, the majority of the features were numerical values. Therefore, to take both these factors into account a combined score from PCC and Chi-square test was used. The scores from the Chi-square tests shown in tables 9.3 and 9.4 were first normalized to a range of [0,1] before combining with PCC to obtain tables 4.5 and 9.4. The top most features from these tables were considered to be the most impactful features in determining a specific class.

$$X_c^2 = \Sigma \frac{(O_i - E_i)^2}{E_i} \tag{4.2}$$

Table 4.5: Ranking based on both PCC and Chi-square test for binary classification

Feature	Absolute value of PCC	Normalized value of Chi-square test	Total score
SrcTCPBase	0.023536	1.000000e+00	1.023536
SrcLoad	0.017765	7.686946e-01	0.786459
TotBytes	0.313687	4.105528e-01	0.724240
SrcBytes	0.382140	2.445810e-01	0.626721
sMeanPktSz	0.586275	4.213180e-04	0.586696
dTtl	0.562927	1.380461e-06	0.562928
CON	0.530657	1.561801e-06	0.530659
SrcPkts	0.409042	9.511207e-05	0.409137
sTtl	0.402824	6.824088e-07	0.402825
TotPkts	0.313170	1.984701e-04	0.313369
Load	0.002067	3.090770e-01	0.311144
dMeanPktSz	0.309784	4.005733e-04	0.310185
dHops	0.277891	1.125245e-07	0.277891
udp	0.171260	1.226915e-07	0.171260
e r	0.166214	1.669391e-07	0.166215
e i	0.132709	1.032687e-07	0.132709
sHops	0.121578	5.037728e-07	0.121578
SrcWin	0.115640	1.308016e-03	0.116949
REQ	0.105914	4.940390e-08	0.105914
icmp	0.095541	3.715706e-08	0.095541

Offset	0.049753	4.055701e-02	0.090310
INT	0.079350	3.246522e-08	0.079350
Dur	0.075601	9.182656e-08	0.075601
tcp	0.070020	1.829477e-08	0.070020
e s	0.054293	1.571344e-08	0.054293
RST	0.053286	1.515984e-08	0.053286
ACC	0.048298	1.326219e-08	0.048298
SrcLoss	0.044073	1.302477e-08	0.044073
DstWin	0.041482	5.336539e-04	0.042015
Loss	0.039236	1.441908e-08	0.039236
e g	0.037960	8.431214e-09	0.037960
RSP	0.035984	8.734909e-09	0.035984
URP	0.032955	7.861389e-09	0.032955
FIN	0.031316	5.919000e-09	0.031316
e &	0.028903	4.891424e-09	0.028903
IdleTime	0.028158	7.310063e-10	0.028158
TcpRtt	0.024803	1.158371e-09	0.024803
AckDat	0.023935	1.046364e-09	0.023935
e d	0.013665	1.153915e-09	0.013665
SynAck	0.011245	2.256173e-10	0.011245
CLO	0.011194	1.047939e-09	0.011194
DstTCPBase	0.002806	8.008176e-03	0.010814
Rate	0.009414	9.413640e-04	0.010355
DstLoss	0.010298	2.415295e-09	0.010298
nan	0.008086	3.832500e-10	0.008086
DstRate	0.005708	1.992057e-04	0.005907
e *	0.004907	1.586530e-10	0.004907
e	0.004386	2.233556e-10	0.004386
SRC	0.004385	1.127874e-10	0.004385
ECR	0.002294	3.352372e-11	0.002294
PAR	0.001601	1.448907e-11	0.001601
TST	0.001039	4.545590e-12	0.001039
URH	0.000936	4.545590e-12	0.000936
e s	0.000260	0.000000e+00	0.000260

Table 4.6: Ranking based on both PCC and Chi-square test for multi class classification

Feature	Absolute value of PCC	Normalized value of Chi-square test	Total score
SrcTCPBase	0.095418	1.000000e+00	1.095418
tcp	0.672815	7.550484e-09	0.672815
udp	0.639528	8.393535e-09	0.639528
RST	0.471204	6.983077e-09	0.471204
Load	0.024665	4.067659e-01	0.431431
INT	0.379360	4.925269e-09	0.379360
sHops	0.270401	3.689460e-08	0.270401
dTtl	0.254632	2.997173e-09	0.254632
e s	0.245781	6.787404e-09	0.245781

Loss	0.241873	6.575312e-09	0.241873
SrcLoss	0.235020	7.476499e-09	0.235020
AckDat	0.231549	6.710481e-10	0.231549
TcpRtt	0.222615	4.493688e-10	0.222615
FIN	0.213388	1.474718e-09	0.213388
IdleTime	0.198134	4.153752e-10	0.198134
SrcBytes	0.170545	5.311210e-04	0.171077
dMeanPktSz	0.166248	3.469100e-06	0.166252
ACC	0.160435	3.119457e-09	0.160435
dHops	0.149132	2.483168e-10	0.149132
SrcPkts	0.143269	2.124265e-07	0.143269
DstWin	0.137342	1.290847e-04	0.137471
TotBytes	0.131801	8.956791e-04	0.132697
e d	0.127613	4.189570e-10	0.127613
REQ	0.121778	4.083746e-09	0.121778
DstLoss	0.115615	1.266933e-09	0.115615
sMeanPktSz	0.107542	1.258462e-06	0.107543
TotPkts	0.100205	4.493885e-07	0.100205
sTtl	0.090589	2.360515e-09	0.090589
SynAck	0.084669	3.182591e-10	0.084669
e r	0.076001	3.623512e-10	0.076001
icmp	0.072593	8.747194e-09	0.072593
CON	0.070229	6.517445e-09	0.070229
e *	0.062180	1.277205e-10	0.062180
Offset	0.052609	8.091226e-03	0.060700
SrcWin	0.060693	5.034954e-06	0.060699
e i	0.058434	2.245652e-10	0.058434
Dur	0.040427	8.427850e-09	0.040427
SrcLoad	0.019892	8.316288e-03	0.028208
Rate	0.021532	5.527116e-05	0.021587
DstRate	0.019422	3.466995e-05	0.019457
RSP	0.016510	1.889539e-11	0.016510
URP	0.015120	1.699903e-11	0.015120
e g	0.014919	1.862780e-11	0.014919
e &	0.013261	1.055143e-11	0.013261
TST	0.008047	6.521014e-12	0.008047
URH	0.007558	1.298516e-12	0.007558
CLO	0.005136	2.207460e-12	0.005136
nan	0.003252	4.515457e-11	0.003252
DstTCPBase	0.002969	1.231003e-04	0.003092
SRC	0.001764	1.326417e-11	0.001764
ECR	0.000922	3.918134e-12	0.000922
e s	0.000849	0.000000e+00	0.000849
e	0.000822	4.571247e-13	0.000822
PAR	0.000644	1.673747e-12	0.000644

4.6 Data Normalization

The preprocessing of data is a crucial step in attaining strong classification performance using machine learning models. As one of the primary data preprocessing steps, normalization entails the adjustment of features to a common range such that larger numeric feature values do not predominate smaller numeric feature values [57]. This makes all features in a dataset equally significant through equal numeric contribution in determining the output class.

Many different methods are been proposed to normalize raw data to a specified range through various statistical measures. In this instance, we employed the Z-score scalar for all ML models with the exception of Support Vector Machines (SVM). The Min-max scaler was utilized for the SVM because it accelerated the SVM's learning process.

4.6.1 Z-Score

In Z-Score normalization, the data are rescaled using the mean and standard deviation values so that the resulting features have a zero mean and a unit variance [57]. The equation 4.3 is been used to calculate the normalized values.

$$\hat{x}_{i,n} = \frac{x_{i,n} - \mu_i}{\sigma_i} \tag{4.3}$$

where $\hat{x}_{i,n}$ is the resultant normalized value and $x_{i,n}$ is the original value of n^{th} data in the i^{th} feature column. μ_i is the mean and σ_i is the standard deviation of the i^{th} feature. The Z-Score normalization was performed through the Standard scaler in Scikit learn library.

4.6.2 Min-max Normalization

This technique uses the minimum and maximum values of a specific feature to scale the data to a predetermined range. Usually, the range is specified as [0,1] or [-1,1]. The MinMax Scaler in Scikit learn was used to normalize the values before feeding onto SVM model. The equation 4.4 demonstrate formula used for the calculation of normalized values using this technique.

$$\hat{x}_{i,n} = \frac{x_{i,n} - min(x_i)}{max(x_i) - min(x_i)}(Upperbound - Lowerbound) + Lowerbound \tag{4.4}$$

The $\hat{x}_{i,n}$ represent the normalized value while $x_{i,n}$ is the original value available in the dataset. The maximum and minimum values in a particular feature is represented by $min(x_i)$ and $max(x_i)$. The upper bound and the lower bound specifies the boundaries of the normalized range. In this work, the range was set as [-1,1]

5 RESULTS

5.1 Performance measuring metrics

The performance of different machine learning models were evaluated through several parameters that are been commonly used for evaluation purposes. This includes performance indicators such as precision, recall, F1-score, accuracy along with the confusion matrix. Further, concepts such as ROC and AUC were examined in comparing the machine learning models.

5.1.1 Confusion Matrix

For a classification problem, the performance can be measured in many different techniques depending on the characteristics of data. The performance metrics used for the measurement is mostly computed using a confusion matrix. The confusion matrix is a representation of all the test samples categorized into either True Positive (TP), False Negative (FN), False Positive (FP) or True Negative (TN) for a binary classification. The rows in table 5.1 demonstrate the predicted class after classification and the columns represent the actual class that the sample belongs to. TP and TN reflect the number of correctly categorized positive and negative cases, whereas FP and FN indicate the number of incorrectly classified instances [58].

Table 5.1: Confusion Matrix for binary classification

	Actual Positive Class	Actual Negative Class
Predicted Positive Class	True Positive (TP)	False Negative (FN)
Predicted Negative Class	False Positive (FP)	True Negative (TN)

The confusion matrix for a multi-class classification consists of rows and columns proportional to the number of possible classes. As in the binary classification, the rows reflect the predicted class and the columns represent the actual class of the data. The table 5.2 shows a confusion matrix for a multi class classification between 3 classes. Since the classes are listed in the same order in both rows and columns, the correctly categorized elements are positioned along the diagonal, from the top left to the bottom right [59].

Table 5.2: Confusion Matrix for multi class classification

	Actual			
	Classes	Class 1	Class 2	Class 3
Predicted	Class 1	TP	E_{21}	E_{31}
	Class 2	E_{12}	TP	E_{32}
	Class 3	E_{13}	E_{23}	TP

Here TP has the usual meaning while E is the error in classification which occurs when the predicted class and the actual class does not match.

5.1.2 Precision

The precision is defined as the proportion of true positives to the total number of values that were predicted as positive. Precision measures the accuracy with which a model classifies a test sample as positive. For a classification with multiple classes, precision is determined by having the ratio of the total number of true positives of all classes with the total number of true positives and false positives of all classes.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

5.1.3 Recall

Recall can be computed as the ratio of correctly classified positive samples to the total number of actual positive samples. In a classification problem involving more than two classes, recall is determined by dividing the total number of true positives across all classes by the total number of true positives and false negatives across all classes.

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

5.1.4 F1-Score

The F1-Score is a measurement that takes both Precision and Recall into account. Increasing either precision or recall typically decreases the other measure. In such situations, F1-Score will be a useful metric for determining a model's performance. F1-Score is defined as the harmonic mean or weighted average of recall and precision.

$$F1 - Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{5.3}$$

5.1.5 Accuracy

Accuracy is a measure of number of correctly classified samples to the total number of samples.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5.4}$$

5.1.6 ROC Curves and AUC values

ROC curves are a popular method used for comparison of different classification methods. It is a probability curve with x axis representing the false positive rate (FPR) and y axis representing the true positive rate (TPR). The TPR is the same as recall given in 5.2. The FPR is defined to be the value of $1 - Specificity$ where specificity is defined as in equation 5.5

$$Specificity = \frac{TN}{TN + FP} \tag{5.5}$$

The AUC value which is the Area under the ROC curve represents the degree of separability. To elaborate further, it specifies the model's capacity to differentiate between different classes. The greater the value of AUC, the more accurately the model predicts the proper class. A number closer to 1 indicates that the model performs well in terms of separability [60].

5.2 Model Parameters

Following the processing of data, a some common machine learning models and neural networks were employed to evaluate their performance on the dataset. Decision Tree, Random Forest, K-Nearest Neighbor, Support Vector Machines, and Multi Layer Perceptron as a neural network were among the chosen models. The python scripts were developed using the Scikit learn library. The model parameters for each of the ML models were set to attain maximum accuracy as well as considerably less training times. The model parameters used for each of the classification models are given below under each subsection.

5.2.1 Decision Tree

The scikit learn DecisioTreeClassifier was used in order to model. The criterion parameter was set to "gini" and no maximum depth was specified in order to achieve the maximum accuracy levels. All other parameters were set as default values. The data was normalized using StandardScaler in scikit learn library and the test sample was considered to be 30% of the total data.

5.2.2 Random Forest

RandomForestClassifier from the scikit learn library was used to implement the model. The number of trees was given as 10 in order to obtain the optimum accuracy levels at lower computational time. All other parameters were kept at their default values. In here too, the data was normalized using the Z-score normalization and test sample of 30% of the data was considered.

5.2.3 K-Nearest Neighbor

The K-Nearest Neighbor classifier algorithm from scikit learn library was used. The number of neighbors k was set as 5 in order to have the best possible accuracy along with the optimal time for prediction. The distance metric to use was set as 'minkowski' with a power parameter of 2 which represents the euclidean distance. The algorithm to be used was set to 'auto' so that the most appropriate algorithm based on the values passed will be applied. For the KNN, the number of test samples was set to 0.30 and Z-Score normalization was used.

5.2.4 Support Vector Machines

The Support Vector Classification (SVC) library of scikit learn was used for the implementation of SVM. The regularization parameter C was set to 1 and the kernel was set to linear to achieve a lower training time while obtaining the best accuracy levels. The model was trained with only 20% of the whole data as it performs better with fewer data.

5.2.5 Multi Layer Perceptron

The MLPClassifier in neural network class of the scikit learn library was used to train the model. Three hidden layers was specified with 10,20 and 10 neurons in each to obtain the highest possible accuracy levels.

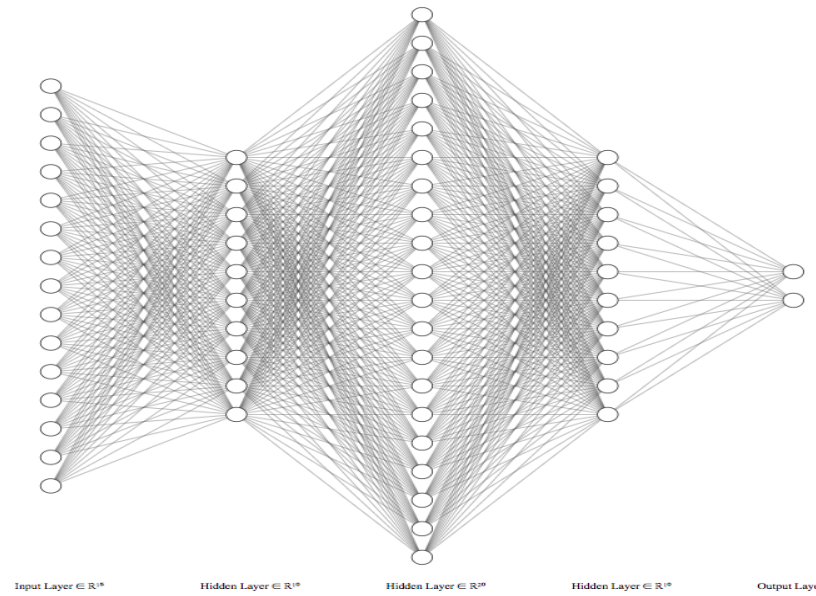


Figure 5.1: MLP Neural Network for Binary classification with 15 input features

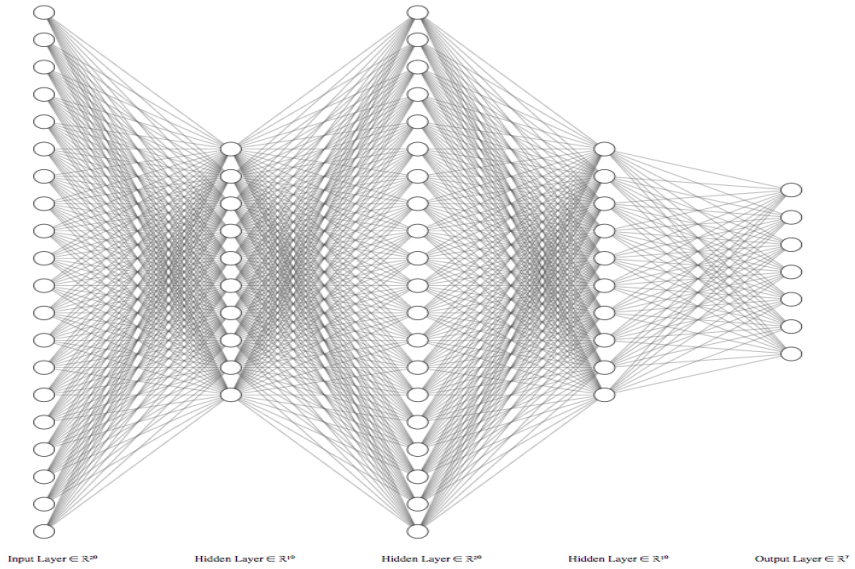


Figure 5.2: MLP Neural Network for Multi-class classification with 20 input features

5.3 Optimal number of feature selection for binary classification

The binary classification was done in order to predict whether a data sample is malicious or not using different machine learning techniques. The results are accumulated for different number of features using the evaluation metrics described in section 5.1. The number of features chosen to evaluate were 5, 10, 15, 20 and 25. The features which contributed significantly to the decision making process was selected as the top most features to aid the classification. These top features were selected using the feature ranking given in table 4.5. The obtained results for different number of features are followed in the next 5 sub sections.

5.3.1 Best 5 features

Table 5.3: Results of different evaluation metrics obtained for binary classification with top 5 features

Model	0/1	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0 1	0.1295 0.9925	0.8583 0.7655	0.2251 0.8643	0.76910033	1.0286	0.0173
Random Forest	0 1	0.1517 0.9912	0.8230 0.8128	0.2561 0.8932	0.81322201	6.6054	0.2021
KNN	0 1	0.7430 0.9912	0.7361 0.8128	0.7396 0.8932	0.97974248	0.4987	899.3506
SVM	0 1	0.9637 0.9893	0.4957 0.9896	0.6547 0.9895	0.97973020	139.0524	535.6866
MLP	0 1	0.3717 0.9875	0.7034 0.9516	0.4864 0.9892	0.94194516	217.9660	0.3910

5.3.2 Best 10 features

Table 5.4: Results of different evaluation metrics obtained for binary classification with top 10 features

Model	0/1	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0 1	0.6000 0.9999	0.9977 0.9730	0.7493 0.9862	0.97391937	1.3565	0.0311
Random Forest	0 1	0.5552 0.9999	0.9981 0.9675	0.7135 0.9834	0.96867975	7.4246	0.2061
KNN	0 1	0.9990 0.9998	0.9952 1.0000	0.9971 0.9999	0.99977287	1.1754	1219.2877
SVM	0 1	0.9740 0.9893	0.7148 0.9896	0.8245 0.9895	0.98819173	270.7133	104.1469
MLP	0 1	0.9987 0.9998	0.9961 0.9999	0.9974 0.9999	0.99979636	53.7057	0.4737

5.3.3 Best 15 features

Table 5.5: Results of different evaluation metrics obtained for binary classification with top 15 features

Model	0/1	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0 1	0.9995 0.9999	0.9981 0.9730	0.9988 0.9862	0.99990601	2.0064	0.0392
Random Forest	0 1	0.9989 0.9999	0.9976 1.0000	0.9982 0.9999	0.99986293	8.2830	0.2314
KNN	0 1	0.9999 0.9999	0.9965 1.0000	0.9982 0.9999	0.99985902	1.7229	1343.3134
SVM	0 1	0.9999 0.9908	0.7702 1.0000	0.8702 0.9954	0.99105532	547.8780	100.9413
MLP	0 1	0.9998 0.9999	0.9972 1.0000	0.9985 0.9999	0.99988251	50.5964	0.3705

5.3.4 Best 20 features

Table 5.6: Results of different evaluation metrics obtained for binary classification with top 20 features

Model	0/1	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0 1	0.9912 1.0000	0.9995 0.9996	0.9953 0.9998	0.9996318	2.4720	0.0428
Random Forest	0 1	0.9966 1.0000	0.9998 0.9999	0.9982 0.9999	0.99985902	10.0376	0.2466
KNN	0 1	0.9989 0.9999	0.9968 1.0000	0.9978 0.9999	0.97974248	0.0761	4725.5620
SVM	0 1	0.9985 0.9916	0.7910 1.0000	0.8827 0.9958	0.99183216	123.6066	99.9657
MLP	0 1	0.9965 1.0000	0.9989 0.9999	0.9977 0.9999	0.99981986	52.0740	0.3156

5.3.5 Best 25 features

Table 5.7: Results of different evaluation metrics obtained for binary classification with top 25 features

Model	0/1	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0 1	0.9933 1.0000	0.9992 0.9997	0.9963 0.9998	0.99970629	4.7324	0.0584
Random Forest	0 1	0.9992 1.0000	0.9989 1.0000	0.9990 1.0000	0.99992559	13.1982	0.2460
KNN	0 1	0.9995 0.9998	0.9944 1.0000	0.9969 0.9999	0.99976112	0.0761	5643.8833
SVM	0 1	0.9599 0.9916	0.9033 1.0000	0.9307 0.9958	0.99480736	2119.9760	92.4467
MLP	0 1	0.9995 0.9999	0.9982 1.0000	0.9988 1.0000	0.99990993	43.2163	0.2338

5.3.6 Comparison of performance with number of features

The analysis of values obtained for the performance evaluation metrics under different number of features show that the overall accuracy levels has increased with the number of features. The Decision Tree model has shown a significant increase in accuracy through the selection of 10 features rather than 5. However, even with 10 features, the Precision and Recall values for detecting benign flows has been lower for Decision Tree as seen in table 5.4. An overall better values for Precision and Recall in Decision Tree can be seen from 15 features and upwards.

The training time and prediction time has seen a slight increase with the increase of number of features in Decision Tree.

A similar phenomena is observed in the Random Forest classifier as well with training times increasing slightly and prediction times staying largely constant. The Random Forest classifier based on number of Decision Trees also fared similar in Precision and Recall values for lesser number of features. An overall satisfactory values for these two evaluation metrics was seen after 15 features.

The K-Nearest Neighbor classifier performed exceptionally well across all evaluation metrics when 10 or more features were used. KNN is distinguished by its shorter training time but extremely lengthy prediction times. In this instance too, despite the fact that the training time for all the samples was very short and around the same, the prediction time has been extremely lengthy and has increased as the number of features has grown.

The results obtained for SVM shows a very high training time compared to other models. Overall, the training time has increased with the increase in number of features except on the instance of using 20 features which was unexpected. In comparison to most other models, considerably good levels of accuracy was obtained even with 5 features. The recall and f1-score for detecting 0 were, however, relatively low. Despite the fact that the values for these parameters increased as the number of features increased, they remained lower than those of other models.

The Multi Layer Perceptron neural network with 3 hidden layers achieved overall good accuracy levels from 10 features upwards. Apart from the occasion where training 5 features took around 217 seconds, the average training time was around 50 seconds on other instances. A drop in the prediction time with the increase of number of features can be seen but it is insignificant in the larger context.

Overall, the results indicate that exceptionally high levels of accuracy could be reached using 15 features for all evaluation measures across all models. As expected, the growing number of features has led to an increase in accuracy across all models. In an IDS, however, both training time and prediction time are crucial measures for an efficient network. Therefore, due to the fact that the training time and prediction time for most models increases as the number of features increases, the optimal number of features to attain a high level of accuracy was determined to be 15.

5.4 Comparison of performance between ML models for binary classification

A comparison of different machine learning models was done in order to identify the suitability of each of the models. For this purpose, the confusion matrices and ROC curves were generated. Tables 5.8, 5.9, 5.10, 5.11 and 5.12 show the confusion matrices obtained for each of the classifiers. The classes 0 and 1 in the confusion matrices represent the benign and malicious traffic flows. Further, the figure 5.3 show the ROC curves and AUC values obtained for all the classifiers.

Table 5.8: Confusion Matrix for Decision Tree at 15 features

	True - 0	True - 1
Predicted - 0	9959	19
Predicted - 1	5	245379

The Decision Tree classifier has performed well with only 19 out of all malicious flows being

undetected. Compared to other models however, both Decision Tree and Random Forest have slightly higher number of false positives. The ROC curves show that the accuracy levels are very close to ideal levels in both Decision Tree and Random Forest classifiers. Random Forest with multiple Decision Tree classifiers has been able to achieve a better AUC score than the Decision Tree classifier. The training time for Random Forest has been little lengthy compared to Decision Tree.

Table 5.9: Confusion Matrix for Random Forest at 15 features

	True - 0	True - 1
Predicted - 0	9954	24
Predicted - 1	11	245373

The KNN classifer has also performed exceptionally well with only 1 flow being identified as a false alarm. In comparison to both Decision Tree and Random Forest, this is an improved performance. This is also reflected in the AUC score on the ROC curve. However, as expected in KNN, the time taken for prediction has been very large.

Table 5.10: Confusion Matrix for KNN at 15 features

	True - 0	True - 1
Predicted - 0	9943	35
Predicted - 1	1	245483

A higher number of network flows in the testing phase of SVM is seen as the test train split was different. However, the confusion matrix for SVM has demonstrates a larger percentage of malicious flows has not been detected. This is clearly seen when the ROC curve is analysed as it goes below the diagonal at lower false positive rates. On the other hand, it has fared very well by keeping the false positive rate to a minimum. As seen from table 5.5 the SVM has taken a very long time to train and also in prediction compared to other models.

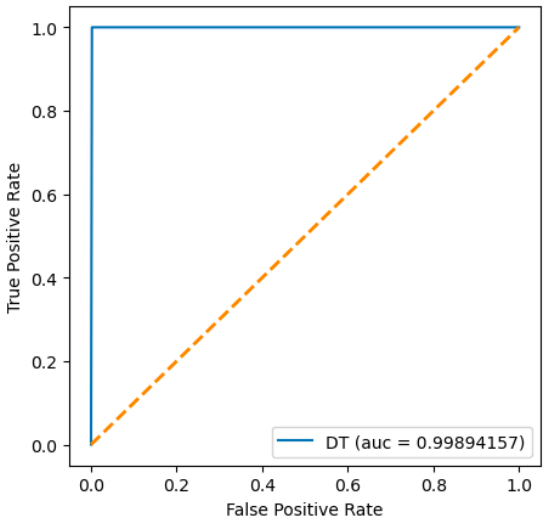
Table 5.11: Confusion Matrix for SVM at 15 features

	True - 0	True - 1
Predicted - 0	20412	6089
Predicted - 1	2	654461

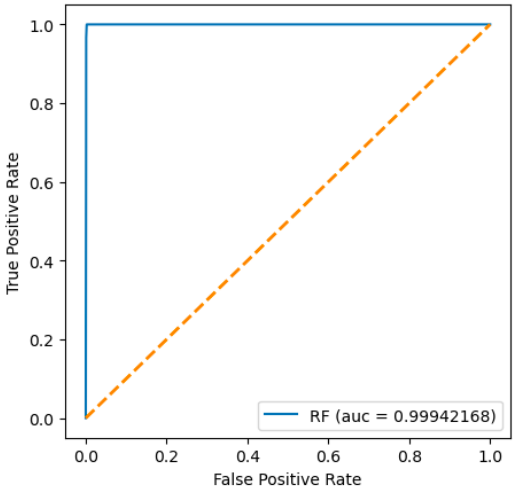
The MLP classifier has fared well with the AUC score being the largest out of all models. However, the number of malicious flows passed undetected has been slightly higher than both Decision Tree and Random Forest. In contrast, it has performed exceptionally well in producing very low false positives. Although, prediction times have been smaller, the training time can be considered average.

Table 5.12: Confusion Matrix for MLP at 15 features

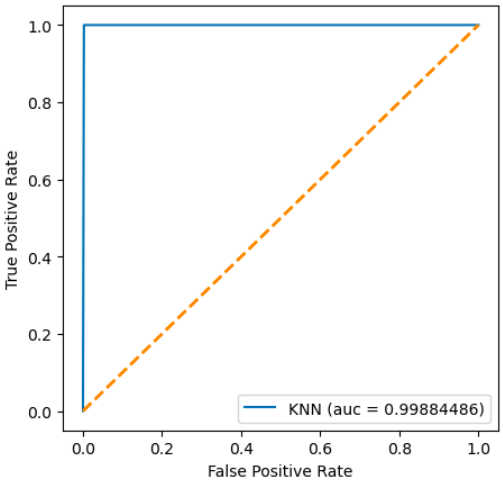
	True - 0	True - 1
Predicted - 0	9950	28
Predicted - 1	2	245382



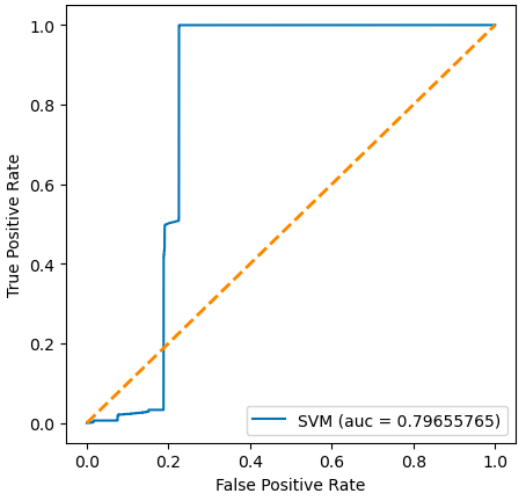
(a) ROC Curve for Decision Tree



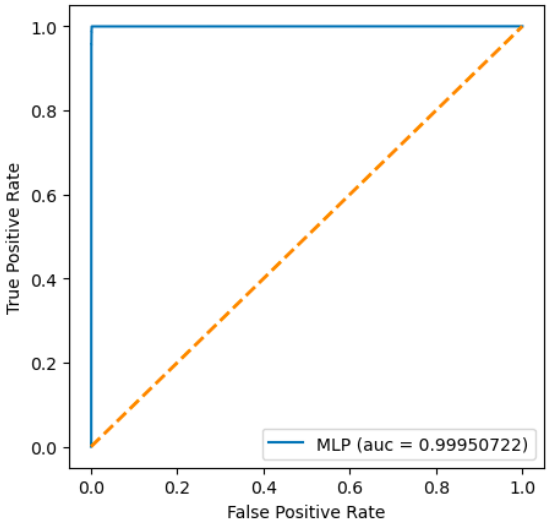
(b) ROC Curve for Random Forest



(c) ROC Curve for K-Nearest Neighbor



(d) ROC Curve for Support Vector Machines



(e) ROC Curve for Multi Layer Perceptron

Figure 5.3: ROC Curves for all ML models

5.5 Optimal number of feature selection for multi class classification

A multi-class classification was conducted to evaluate the levels of accuracy that may be attained when identifying various sorts of attacks. In order to attain homogeneity, the machine learning models utilized for binary classification were also applied here. The ML models were evaluated using the top 5, 10, 15, 20, and 25 features to discover the ideal number of features for achieving high levels of accuracy in the shortest amount of time. The parts that follow in this section will provide both results and a comparative analysis.

5.5.1 Best 5 features

Table 5.13: Results of different evaluation metrics obtained for multi class classification with top 5 features

Model	Class	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0	0.0498	0.8110	0.0938	0.13210266	2.9302	0.0465
	1	0.0000	0.0000	0.0000			
	2	0.0000	0.0000	0.0000			
	3	0.0000	0.0000	0.0000			
	4	0.0000	0.0000	0.0000			
	5	0.2779	0.8902	0.4236			
	6	0.1267	0.0084	0.0157			
Random Forest	0	0.0503	0.8069	0.0947	0.13767122	11.4922	0.4204
	1	0.0000	0.0000	0.0000			
	2	0.0000	0.0000	0.0000			
	3	0.0000	0.0000	0.0000			
	4	0.0000	0.0000	0.0000			
	5	0.2851	0.9432	0.4379			
	6	0.1545	0.0041	0.0079			
KNN	0	0.0497	0.8057	0.0936	0.14101941	0.4377	21.5446
	1	0.1724	0.0005	0.0010			
	2	0.0000	0.0000	0.0000			
	3	0.0000	0.0000	0.0000			
	4	0.7199	0.0290	0.0558			
	5	0.2952	0.9080	0.4455			
	6	0.1649	0.0434	0.0688			
SVM	0	0.0000	0.0000	0.0000	0.81107224	266.0979	1576.1416
	1	0.0000	0.0000	0.0000			
	2	0.8720	1.0000	0.9316			
	3	0.9881	1.0000	0.9940			
	4	0.6437	0.9963	0.7821			
	5	0.5349	0.4636	0.4967			
	6	0.0000	0.0000	0.0000			
MLP	0	0.0894	0.6927	0.1584	0.52167119	267.8803	0.3698
	1	0.6685	0.0620	0.1134			
	2	0.0000	0.0000	0.0000			
	3	0.9878	1.0000	0.9939			
	4	0.7041	0.4898	0.5777			
	5	0.3595	0.7999	0.4961			
	6	0.0000	0.0000	0.0000			

5.5.2 Best 10 features

Table 5.14: Results of different evaluation metrics obtained for multi class classification with top 10 features

Model	Class	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0	0.9995	0.8892	0.9411	0.84428380	2.8585	0.0453
	1	0.0082	0.0008	0.0015			
	2	0.9565	1.0000	0.9778			
	3	1.0000	1.0000	1.0000			
	4	0.9906	0.6699	0.7993			
	5	0.4464	0.9487	0.6071			
	6	0.3363	0.0748	0.1243			
Random Forest	0	0.9344	0.9534	0.9438	0.50228694	10.7951	0.4070
	1	0.0077	0.0007	0.0013			
	2	0.9655	1.0000	0.9824			
	3	0.0000	0.0000	0.0000			
	4	0.9901	0.4906	0.6561			
	5	0.1960	0.9661	0.3259			
	6	0.0739	0.0566	0.0641			
KNN	0	0.9979	0.9899	0.9939	0.90033756	0.9209	1338.7454
	1	0.8653	0.7014	0.7748			
	2	1.0000	1.0000	1.0000			
	3	1.0000	1.0000	1.0000			
	4	0.8957	0.8286	0.8599			
	5	0.6466	0.8415	0.7313			
	6	0.3234	0.2410	0.2762			
SVM	0	0.8871	0.6329	0.7388	0.90190083	161.4029	1041.1428
	1	0.8921	06760	0.7692			
	2	0.9701	1.0000	0.9848			
	3	0.9998	0.9930	0.9964			
	4	0.7864	0.9962	0.8790			
	5	0.8508	0.7006	0.7684			
	6	0.4446	0.3176	0.3705			
MLP	0	0.9997	0.9900	0.9948	0.92565847	204.1593	0.4191
	1	0.9884	0.6858	0.8098			
	2	1.0000	1.0000	1.0000			
	3	1.0000	1.0000	1.0000			
	4	0.8214	0.9977	0.9010			
	5	0.8465	0.7442	0.7921			
	6	0.4870	0.3561	0.4114			

5.5.3 Best 15 features

Table 5.15: Results of different evaluation metrics obtained for multi class classification with top 15 features

Model	Class	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0	0.3148	0.8935	0.4655	0.84668039	2.5443	0.0558
	1	0.2094	0.0637	0.0977			
	2	0.9565	1.0000	0.9778			
	3	1.0000	1.0000	1.0000			
	4	0.9988	0.6739	0.8048			
	5	0.6957	0.9120	0.7893			
	6	0.3855	0.1380	0.2032			
Random Forest	0	0.0966	0.9614	0.1755	0.55332038	9.0797	0.3796
	1	0.6613	0.5211	0.5829			
	2	1.0000	1.0000	1.0000			
	3	0.0000	0.0000	0.0000			
	4	0.9984	0.6738	0.8046			
	5	0.5537	0.9141	0.6897			
	6	0.7315	0.1204	0.2068			
KNN	0	0.9987	0.9881	0.9934	0.97267017	1.5253	2557.4026
	1	0.9737	0.8323	0.8974			
	2	1.0000	1.0000	1.0000			
	3	1.0000	1.0000	1.0000			
	4	0.9625	0.9984	0.9801			
	5	0.9192	0.9053	0.9122			
	6	0.8003	0.8127	0.8064			
SVM	0	0.9103	0.6332	0.7469	0.93868545	132.8353	1122.0557
	1	0.9387	0.6894	0.7949			
	2	0.9700	1.0000	0.9848			
	3	0.9998	0.9929	0.9964			
	4	0.9416	0.9970	0.9685			
	5	0.8238	0.8924	0.8567			
	6	0.6548	0.5996	0.6260			
MLP	0	1.0000	0.9898	0.9949	0.97516466	388.7545	0.4038
	1	0.9789	0.8363	0.9020			
	2	1.0000	1.0000	1.0000			
	3	1.0000	1.0000	1.0000			
	4	0.9625	1.0000	0.9809			
	5	0.8916	0.9607	0.9249			
	6	0.9054	0.7296	0.8081			

5.5.4 Best 20 features

Table 5.16: Results of different evaluation metrics obtained for multi class classification with top 20 features

Model	Class	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0	0.9992	0.9887	0.9939	0.97088055	2.5026	0.0637
	1	0.8441	0.3121	0.4557			
	2	1.0000	1.0000	1.0000			
	3	1.0000	0.9934	0.9967			
	4	0.9275	0.9985	0.9617			
	5	0.9990	0.9984	0.9987			
	6	0.8033	0.9995	0.8907			
Random Forest	0	0.7184	0.9965	0.8349	0.97415825	8.1971	0.3779
	1	0.8985	0.5036	0.6454			
	2	0.9819	1.0000	0.9908			
	3	1.0000	0.9934	0.9967			
	4	1.0000	0.9996	0.9998			
	5	0.9835	0.9752	0.9794			
	6	0.9665	0.9622	0.9643			
KNN	0	0.9991	0.9960	0.9975	0.99671838	0.0829	4408.4057
	1	0.9981	0.9965	0.9973			
	2	1.0000	1.0000	1.0000			
	3	1.0000	1.0000	1.0000			
	4	0.9990	1.0000	0.9995			
	5	0.9815	0.9915	0.9865			
	6	0.9824	0.9602	0.9712			
SVM	0	0.9890	0.8852	0.9342	0.97291339	59.9453	575.5416
	1	0.9867	0.8131	0.8915			
	2	0.9991	1.0000	0.9996			
	3	0.9954	1.0000	0.9996			
	4	0.9557	1.0000	0.9774			
	5	0.9120	0.9431	0.9273			
	6	0.8783	0.8214	0.8489			
MLP	0	0.9992	0.9966	0.9979	0.99967497	463.5667	0.5153
	1	0.9978	0.9982	0.9980			
	2	1.0000	1.0000	1.0000			
	3	1.0000	1.0000	1.0000			
	4	0.9998	1.0000	0.9999			
	5	0.9993	0.9996	0.9994			
	6	0.9982	0.9985	0.9984			

5.5.5 Best 25 features

Table 5.17: Results of different evaluation metrics obtained for multi class classification with top 25 features

Model	Class	Precision	Recall	F1-Score	Accuracy	Training Time (s)	Prediction Time (s)
Decision Tree	0	0.7737	0.9866	0.8672	0.98612166	2.9903	0.0515
	1	0.9373	0.9993	0.8672			
	2	0.9999	1.0000	0.9999			
	3	1.0000	0.9934	0.9967			
	4	1.0000	0.9393	0.9687			
	5	0.9998	0.9994	0.9996			
	6	0.9999	0.9988	0.9993			
Random Forest	0	0.8425	0.9614	0.8980	0.97640995	7.6865	0.3904
	1	0.8763	0.9974	0.9330			
	2	0.9999	1.0000	1.0000			
	3	1.0000	0.9934	0.9967			
	4	0.9996	0.9637	0.9813			
	5	0.9113	0.9956	0.9516			
	6	0.9934	0.7497	0.8545			
KNN	0	0.9995	0.9953	0.9974	0.99758382	0.0902	5769.9772
	1	0.9989	0.9979	0.9984			
	2	1.0000	1.0000	1.0000			
	3	1.0000	1.0000	1.0000			
	4	0.9992	1.0000	0.9996			
	5	0.9861	0.9941	0.9901			
	6	0.9876	0.9707	0.9791			
SVM	0	0.9963	0.8558	0.9207	0.97205872	110.5324	1569.6127
	1	0.9889	0.8127	0.8922			
	2	0.9990	1.0000	0.9995			
	3	0.9952	1.0000	0.9976			
	4	0.9555	1.0000	0.9773			
	5	0.9089	0.9436	0.9259			
	6	0.8664	0.8255	0.8455			
MLP	0	0.9996	0.9959	0.9977	0.99855890	461.1033	0.7032
	1	0.9979	0.9959	0.9977			
	2	1.0000	1.0000	1.0000			
	3	1.0000	1.0000	1.0000			
	4	0.9998	1.0000	0.9999			
	5	0.9960	0.9927	0.9944			
	6	0.9833	0.9919	0.9875			

5.5.6 Comparison of performance with number of features

The general expectation of increased accuracy scores resulting through increased number of features could be seen in this scenario as well. However, the relationship between the training times and prediction times with increasing number of features was not consistent at all times.

The Decision Tree algorithm has shown a great improvement in accuracy levels from 5 to 10 features and then from 15 to 20 features. The training times and prediction times have been largely constant throughout all the instances. The Decision Tree algorithm has been able to achieve considerable levels of precision, recall and f1-score at around 20 features across all the classes.

The figures have been similar in Random Forest classifier with increase in number of features showing a considerable increase in evaluation metrics. Compared to the Decision Tree classifier, RF has not fared very well under 10 and 15 features. The evaluation metrics have been considerably greater with 20 features and above and has largely been constant thereafter. A major change is not seen with training and prediction times as they have been relatively constant throughout different number of features.

The typically higher prediction times were observed in this instance as well with the KNN classifier. The general trend of increase in prediction times was seen with training times being quite short and constant. The KNN classifier was able to achieve considerably good values for evaluation metrics from 15 features onward while achieving excellent results at 20 features.

The SVM classifier achieved significant levels of accuracy from a very small number of features compared to others. However, the overall values for all evaluation metrics was at a satisfactory level only at around 20 features. The training and prediction times were also comparatively lower with 20 features. A highlight with SVM classifier was that the training time decreased in general with the increase in number of features.

The MLP classifier was capable to achieve considerably good scores at 15 features while improving further to attain excellent outcomes at 20 features. The accuracy levels at 20 and 25 features did not have significant change as well as the training times and prediction times.

The overall results show that the selection of 20 features is the most optimal option in terms of the scores across all evaluation metrics, training times and prediction times.

5.6 Comparison of performance between ML models for multi class classification

The most optimal number of features to achieve a high level of accuracy in all models was considered to be 20. In order to compare the performance of machine learning models, further analysis was done by evaluating the confusion matrices for each of the attacks. The tables 5.18, 5.19, 5.20, 5.21 and 5.22 shows the confusion matrices for each of the classifiers. The classes 0, 1, 2, 3, 4, 5, and 6 represents the benign flows, port scans, UDP flood, ICMP flood, SYN flood, HTTP flood and Slowrate DoS respectively.

The confusion matrix for Decision Tree shows that, it has been able to achieve almost 0 levels of false alarms. Similarly, all UDP flood and ICMP flood flows have been identified correctly. A higher percentage of false classifications can be seen for slowrate DoS attacks, port scans and SYN flood attacks. The overall values obtained for evaluation metrics at 20 features also produced good results except for port scans and slowrate DoS attacks which produced lower levels of Recall and Precision scores.

Table 5.18: Confusion Matrix for Decision Tree multi class classification with 20 features

	True - 0	True - 1	True - 2	True - 3	True - 4	True - 5	True - 6
Predicted - 0	9865	31	0	0	28	28	34
Predicted - 1	1	3027	0	0	3587	0	3084
Predicted - 2	0	0	67352	0	0	0	0
Predicted - 3	0	528	0	79323	0	0	0
Predicted - 4	0	0	0	0	46696	0	72
Predicted - 5	1	0	0	0	44	26637	0
Predicted - 6	0	0	0	0	0	0	13026

The Random Forest classifier has not fared very well with predicting benign traffic as very high amounts of false alarms are observed. The classifier has performed considerably well in detecting ICMP flood attacks and SYN flood attacks. Higher levels of inaccuracies were seen in classification of benign traffic, port scans, UDP flood, HTTP flood and Slowrate DoS attacks.

Table 5.19: Confusion Matrix for Random Forest multi class classification with 20 features

	True - 0	True - 1	True - 2	True - 3	True - 4	True - 5	True - 6
Predicted - 0	9943	4	30	0	0	1	0
Predicted - 1	3598	4884	1215	0	0	0	2
Predicted - 2	0	0	67352	0	0	0	0
Predicted - 3	0	528	0	79323	0	0	0
Predicted - 4	15	0	0	0	46750	3	0
Predicted - 5	266	11	0	0	0	27972	433
Predicted - 6	19	9	0	0	0	465	12539

The KNN classifier has shown an overall good performance for all the classes except for detection of HTTP flood and Slowrate DoS attacks. A considerable portion of HTTP flood flows was seen classified as Slowrate DoS flows while it is same for vice-versa. In terms of time, a higher prediction time with KNN is a challenge for it to be deployed for IDS.

Table 5.20: Confusion Matrix for KNN multi class classification with 20 features

	True - 0	True - 1	True - 2	True - 3	True - 4	True - 5	True - 6
Predicted - 0	9938	3	2	0	19	14	2
Predicted - 1	6	9665	0	0	6	8	14
Predicted - 2	0	0	67352	0	0	0	0
Predicted - 3	0	0	0	79851	0	0	0
Predicted - 4	0	0	0	0	46768	0	0
Predicted - 5	3	10	0	0	24	28437	208
Predicted - 6	0	5	0	0	0	514	12513

The SVM multi class classifier has not performed exceptionally well as seen with the binary classifier also. Comparatively it has shown good performance in detecting UDP flood attacks

and benign flows. However, in comparison to the previous models, the overall performance could be described as inadequate.

Table 5.21: Confusion Matrix for SVM multi class classification with 20 features

	True - 0	True - 1	True - 2	True - 3	True - 4	True - 5	True - 6
Predicted - 0	23359	224	153	982	610	822	237
Predicted - 1	64	20974	0	4	4694	40	20
Predicted - 2	0	0	178649	0	0	0	0
Predicted - 3	0	0	0	213472	0	0	0
Predicted - 4	0	0	0	0	124894	0	0
Predicted - 5	3	162	19	0	8	478	72564
Predicted - 6	34	39	0	3	6	6139	28607

The Multi Layer Perceptron classifier in multi class classification has shown excellent results. The model has been able to achieve almost zero false classifications for all the attack types. However, compared to Decision Tree and Random Forest models, the amount of time taken for training process has been quite longer. Therefore, the selection or application of the ML model for any IDS system would depend on the performance metrics required in the specific network.

Table 5.22: Confusion Matrix for MLP multi class classification with 20 features

	True - 0	True - 1	True - 2	True - 3	True - 4	True - 5	True - 6
Predicted - 0	9944	18	0	2	8	6	0
Predicted - 1	0	9682	0	0	0	0	17
Predicted - 2	0	0	67352	0	0	0	0
Predicted - 3	0	0	0	79851	0	0	0
Predicted - 4	0	0	0	0	46768	0	0
Predicted - 5	3	3	0	0	0	28670	6
Predicted - 6	5	0	0	0	0	15	13012

6 DISCUSSION

This research was conducted to provide a high-quality labeled dataset with a variety of attack types in a real 5G environment, which has long been unavailable to the 5G security research community. The objective of the thesis was achieved through the generation of a dataset in both packet based and flow based formats with all the flows labeled to identify malicious flows as well as different attack types. In addition to this a subsequent analysis was done with some general machine learning models to evalaute the performance of the dataset. This discussion will provide a detailed analysis on the work carried out in building the dataset by comparing it to past works and a descriptive analysis of the results presented in Chapter 5.

6.1 Comparison with other datasets

As described in Chapter 2, numerous datasets have been created for the aim of intrusion detection over the years, with varying degrees of progress. Popular datasets among them, such as KDDCUP99 and DARPA, have proven highly valuable for AI/ML-based intrusion detection research. However, these datasets are extremely outdated and do not reflect the traffic of modern networks, rendering them irrelevant in the current setting. Recent published datasets, such as the BoT-IoT dataset and CAIDA2017, have solved this issue to some extent. However, it is believed that the traffic flows in a real 5G environment will differ due to the merging of numerous technologies as well as high speeds. In light of the growing global deployment of 5G, the security research community needs specific datasets to evaluate their security solutions on 5G networks. This dataset addresses the lack of a 5G-specific dataset by providing a comprehensive dataset involving mobile users and MEC.

In addition to the novel aspect of generating a 5G dataset for the first time, our effort addresses a number of deficiencies in previous datasets. A comprehensive review of the vast majority of datasets reveals that the data were acquired either from virtualized networks through simulation or from networks created solely for the purpose of data collection. More often than not this does not typically provide a complete picture of network traffic in the actual world. In contrast the data collection in this work is carried out from a fully functional 5G network at University of Oulu.

Furthermore, the development of benign traffic is regarded as an essential part of any dataset. Various datasets have utilized different methods to perform the addition of considerable amounts of benign traffic into the network. The production of benign traffic through simulators and the inclusion of previously obtained user profiles into the network at the time of data collection were approaches frequently seen. In distinction, this study explored the possibility of producing traffic in real time from genuine mobile users connected to base stations. We feel this contributes to the enhancement in resembling a real network of mobile users.

This dataset is further distinguished by the availability of data from numerous network nodes. As 5G evolves into a network comprised of a high number of networked devices and integrating technologies, attackers will be able to identify routes with less security mechanisms implemented. Therefore, a global model of coordinated security is needed throughout the network. In the context of 5G, this can be achieved through federated learning based security implementations. Federated learning based security research is been carried out extensively in this age but more often than not the datasets allocated for different workers are splits of the same dataset. The availability of data collected from both base stations in the network will make this dataset useful for federated learning based research in 5G networks too.

Therefore, not limiting to the aspect of uniqueness gained through involvement of 5G, we have addressed limitations in general datasets as well through this work.

6.2 Further analysis of Results

The subsequent testing and validation of the dataset was done through application of common ML classifiers to classify the network flows as a binary classification as well as a multi class classification. The binary classification involved identification of flows as malicious or non malicious while the multi class classification was performed to identify each type of attack.

The binary classification showed that several classification models outperformed others in terms of the accuracy, precision, recall, f1-score and training, prediction times. verall accuracy scores of more than 0.99 have been achieved by each of the five ML models for the top 15 characteristics. The Decision Tree model has demonstrated the highest overall accuracy despite requiring minimal training time. However, compared to other models, the Recall and F1-score for detecting malicious traffic have been lower, as depicted in the table 5.5.

The Random Forest binary classifier with 15 features outperformed the Decision Tree classifier, despite the accuracy score suggesting otherwise. Random Forest provides superior precision, recall, and f1-score values overall. This is demonstrated when the numerical AUC scores of Decision Tree and Random Forest are compared. The lower training times and prediction times is a major feature that is highlighted in both Decision Tree and Random Forest classifiers rather than other classifiers.

As expected, the KNN algorithm has produced extremely short training times. However, the prediction time is extremely lengthy, which may not be optimal in an IDS scenario. The accuracy levels have been extremely good with all evaluation metrics for determining malicious traffic achieving scores very close or equal to one.

The SVM binary classifier has taken a quite a long time for training with only a 20% of data. The accuracy levels achieved by SVM with only 20% of training data is commendable. The recall and f1-score for detecting benign traffic has been very low when compared to other models. This is reflected in the comparatively very low AUC score. This could have been improved with more data, but the time consumption for training the model makes it not the most optimal choice.

The MLP classifier has fared extremely well for precision, recall and f1-score in detecting both malicious and non malicious flows. The training times have been comparatively more with regard to some other models. Out of all classification models, MLP has been able to achieve the highest AUC score.

When the overall context is taken into account, MLP, KNN and Random Forest have generally fared well in terms of all evaluation metrics. However, as speed and latency plays a major role in advanced 5G networks, the intrusion detection needs to be in real-time such that it would not be a bottleneck for the network. Taking these aspects into account, the Random Forest and MLP classifier stands out against other models. However, the application of the classifier is totally dependant on the requirement of the intrusion detection system.

In the multi-class scenario, the confusion matrices demonstrated that different ML models are capable of delivering adequate levels of accuracy in detecting various types of attacks. Both the Decision Tree and the Random Forest classifiers have shown that there is great room for improvement in the prediction of class 1 which corresponds to Port scans. Further, the precision scores for detection of slowrate DoS attacks has been lower for Decision Tree while the precision value for the detection of non-malicious traffic has been lower in the case of Random Forest classifier.

The KNN classifier has fared very well on all the evaluation metrics. However, the larger prediction times is always a drawback with the KNN classifier in a real-time system. Disregarding this, it has performed well in efficient detection of all attack types. SVM on the other hand, has also produced moderate accuracy levels but have not fared well in detecting certain types of

attacks. The performance has been considerably low in detecting attacks such as Slowrate DoS, HTTP flood etc as observed in the confusion matrix shown in table 5.21. Moreover, low recall scores are been seen for both normal traffic and traffic from port scans.

'MLP classifier with 10,20 and 10 neurons each in 3 hidden layers has outperformed most of the other ML models. It has been able to achieve excellent scores across all the evaluation metrics for all attack classes as seen from the confusion matrix in table 5.22. However, the training time of the model has been longer than all other ML models while prediction time being quite small.

In general the observations in the confusion matrices show that there has been difficulties in differentiating between the HTTP flood attacks and Slowrate DoS attacks. The ML models Decision Tree, Random Forest, and KNN has performed well in detecting some of the specific attacks which emphasize that not all models suit all types of attacks. However, MLP has shown to be accurate in detecting all types of attacks.

In terms of the overall average accuracy levels both MLP and KNN have shown excellent results in detecting all attack types. However, both of them have drawbacks in training time and prediction time respectively. On the other hand, Decision trees and Random Forests have been robust in terms of time while achieving considerable levels of accuracy. However, the lower evaluation metrics recorded for some of the attack types constitute a major limitation. Therefore, the selection of best ML model will be based on the requirements in the specific intrusion detection environment. However, a more sophisticated and optimized neural networks may be helpful in achieving higher accuracy with lesser time.

7 CONCLUSION

This dataset was created to give the research community with a well-defined and high-quality dataset for intrusion detection in 5G networks, which had previously been unavailable. Aside from using a fully operational 5G network for data collection, this dataset has overcome several of the recognized drawbacks in earlier datasets. The network architecture is fully defined with attack network and target network isolated from each other. This facilitated in carrying out attacks typically seen in modern networks from attackers outside of the network. Moreover, the benign traffic is created in real time by actual mobile users rather than agents or simulators. The data collection has also been carried out at multiple points in the network creating a large and diverse dataset. All these factors have combined to improve the environment of the dataset to mimic real-world scenario.

The dataset contains 851205 network flows, with 365480 and 485726 flows from each pico base station. All of these network flows have been properly labeled in terms of their malicious and non-malicious natures as well as the attack types. Both packet-based and network flow-based versions of the dataset is available. The data collected from each base station are also available independently. Additionally, data acquired during each attack is available individually prior to aggregation.

To assess the quality of data, the dataset is analysed through popular machine learning models. Before, the application of these models, data was processed in different steps. A feature selection was performed to identify the best features which contribute to the predictions. The ML models were applied on the dataset and tests were carried out for different number of features to determine the optimal number of features required to achieve considerable levels of accuracy.

Several common attack scenarios were executed during this study to generate a diverse dataset. As port scans and DoS/DDoS assaults are pervasive across a range of network setups, they dominated the focus of this particular dataset. Moreover, 5G networks are known to be susceptible to these main types of attacks and the occurrence of them are known to be common in a general network setting. The architecture of our network had to be also taken into account when choosing the type of attacks. To accurately reflect a real-world scenario of an attack, it was crucial that the attackers' network and the victims' network to be isolated from one another. This restriction hindered the execution of some kinds of attacks. However, variety of attacks of port scans and DoS/DDoS can be seen as present in this work. Although this dataset presents a valuable content for the research community in 5G AI based security implementation, this could be further developed through addition of some modern attack types with alterations in the network if required.

This attack dataset can also be used for federated learning based security solutions as well since data has been collected from multiple points in the network. However, during this first attempt of dataset generation from this test bed, data was collected from only two points specifically at two base stations. This can be developed to collect data from other points of the network too such as the MEC, switch etc. This will generate specific datasets which can be useful in application and testing of federated learning based security solutions.

8 BIBLIOGRAPHY

[1] Salameh A.I. & El Tarhuni M. (2022) From 5g to 6g—challenges, technologies, and applications. Future Internet 14, pp. 117.

[2] Attaran M. (2021) The impact of 5g on the evolution of intelligent automation and industry digitization. Journal of Ambient Intelligence and Humanized Computing pp. 1–17.

[3] Bhalla M.R., Bhalla A.V. et al. (2010) Generations of mobile wireless technology: A survey. International Journal of Computer Applications 5, pp. 26–32.

[4] Wang C.X., Haider F., Gao X., You X.H., Yang Y., Yuan D., Aggoune H.M., Haas H., Fletcher S. & Hepsaydir E. (2014) Cellular architecture and key technologies for 5g wireless communication networks. IEEE communications magazine 52, pp. 122–130.

[5] Jiang W., Han B., Habibi M.A. & Schotten H.D. (2021) The road towards 6g: A comprehensive survey. IEEE Open Journal of the Communications Society 2, pp. 334–366.

[6] Letaief K.B., Chen W., Shi Y., Zhang J. & Zhang Y.J.A. (2019) The roadmap to 6g: Ai empowered wireless networks. IEEE communications magazine 57, pp. 84–90.

[7] Ahmad I., Kumar T., Liyanage M., Okwuibe J., Ylianttila M. & Gurtov A. (2018) Overview of 5g security challenges and solutions. IEEE Communications Standards Magazine 2, pp. 36–43.

[8] Ahmad I., Shahabuddin S., Kumar T., Okwuibe J., Gurtov A. & Ylianttila M. (2019) Security for 5g and beyond. IEEE Communications Surveys & Tutorials 21, pp. 3682–3722.

[9] Liyanage M., Porambage P., Ding A.Y. & Kalla A. (2021) Driving forces for multi-access edge computing (mec) iot integration in 5g. ICT Express 7, pp. 127–137.

[10] Gharib A., Sharafaldin I., Lashkari A.H. & Ghorbani A.A. (2016) An evaluation framework for intrusion detection dataset. In: *2016 International Conference on Information Science and Security (ICISS)*, IEEE, pp. 1–6.

[11] Haider N., Baig M.Z. & Imran M. (2020) Artificial intelligence and machine learning in 5g network security: Opportunities, advantages, and future research trends. arXiv preprint arXiv:2007.04490 .

[12] Tsai C.F., Hsu Y.F., Lin C.Y. & Lin W.Y. (2009) Intrusion detection by machine learning: A review. expert systems with applications 36, pp. 11994–12000.

[13] Khraisat A., Gondal I. & Vamplew P. (2018) An anomaly intrusion detection system using c5 decision tree classifier. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, pp. 149–155.

[14] Khraisat A., Gondal I., Vamplew P. & Kamruzzaman J. (2019) Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2, pp. 1–22.

[15] Saranya T., Sridevi S., Deisy C., Chung T.D. & Khan M.A. (2020) Performance analysis of machine learning algorithms in intrusion detection system: a review. Procedia Computer Science 171, pp. 1251–1260.

[16] Qu H., Qiu Z., Tang X., Xiang M. & Wang P. (2018) Incorporating unsupervised learning into intrusion detection for wireless sensor networks with structural co-evolvability. Applied Soft Computing 71, pp. 939–951.

[17] Van Engelen J.E. & Hoos H.H. (2020) A survey on semi-supervised learning. Machine Learning 109, pp. 373–440.

[18] Rutkowski L., Jaworski M., Pietruczuk L. & Duda P. (2013) Decision trees for mining data streams based on the gaussian approximation. IEEE Transactions on Knowledge and Data Engineering 26, pp. 108–119.

[19] Tan X., Su S., Huang Z., Guo X., Zuo Z., Sun X. & Li L. (2019) Wireless sensor networks intrusion detection based on smote and the random forest algorithm. Sensors 19, pp. 203.

[20] Lin W.C., Ke S.W. & Tsai C.F. (2015) Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. Knowledge-based systems 78, pp. 13–21.

[21] Reddy E.K. (2013) Neural networks for intrusion detection and its applications. In: *Proceedings of the World Congress on Engineering*, volume 2, pp. 3–5.

[22] Catania C.A. & Garino C.G. (2012) Automatic network intrusion detection: Current techniques and open issues. Computers & Electrical Engineering 38, pp. 1062–1072.

[23] Esmaily J., Moradinezhad R. & Ghasemi J. (2015) Intrusion detection system based on multi-layer perceptron neural networks and decision tree. In: *2015 7th Conference on Information and Knowledge Technology (IKT)*, IEEE, pp. 1–5.

[24] Ring M., Wunderlich S., Scheuring D., Landes D. & Hotho A. (2019) A survey of network-based intrusion detection data sets. Computers & Security 86, pp. 147–167.

[25] Creech G. & Hu J. (2013) A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. IEEE Transactions on Computers 63, pp. 807–819.

[26] McHugh J. (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Transactions on Information and System Security (TISSEC) 3, pp. 262–294.

[27] Thakkar A. & Lohiya R. (2020) A review of the advancement in intrusion detection datasets. Procedia Computer Science 167, pp. 636–645.

[28] Nehinbe J.O. (2011) A critical evaluation of datasets for investigating idss and ipss researches. In: *2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)*, IEEE, pp. 92–97.

[29] Sangster B., O'Connor T., Cook T., Fanelli R., Dean E., Morrell C. & Conti G.J. (2009) Toward instrumenting network warfare competitions to generate labeled datasets. In: *CSET*.

[30] Song J., Takakura H., Okabe Y., Eto M., Inoue D. & Nakao K. (2011) Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. In: *Proceedings of the first workshop on building analysis datasets and gathering experience returns for security*, pp. 29–36.

[31] Sperotto A., Sadre R., Vliet F.v. & Pras A. (2009) A labeled data set for flow-based intrusion detection. In: *International Workshop on IP Operations and Management*, Springer, pp. 39–50.

[32] Shiravi A., Shiravi H., Tavallaee M. & Ghorbani A.A. (2012) Toward developing a systematic approach to generate benchmark datasets for intrusion detection. computers & security 31, pp. 357–374.

[33] Xie M. & Hu J. (2013) Evaluating host-based anomaly detection systems: A preliminary analysis of adfa-ld. In: *2013 6th international congress on image and signal processing (CISP)*, IEEE, volume 3, pp. 1711–1716.

[34] Moustafa N. & Slay J. (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: *2015 military communications and information systems conference (MilCIS)*, IEEE, pp. 1–6.

[35] Koroniotis N., Moustafa N., Sitnikova E. & Turnbull B. (2019) Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Generation Computer Systems 100, pp. 779–796.

[36] Sharafaldin I., Lashkari A.H. & Ghorbani A.A. (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp 1, pp. 108–116.

[37] Jazi H.H., Gonzalez H., Stakhanova N. & Ghorbani A.A. (2017) Detecting http-based application layer dos attacks on web servers in the presence of sampling. Computer Networks 121, pp. 25–36.

[38] Wei Y., Zhou S., Leng S., Maharjan S. & Zhang Y. (2021) Federated learning empowered end-edge-cloud cooperation for 5g hetnet security. IEEE Network 35, pp. 88–94.

[39] Bhuyan M.H., Bhattacharyya D.K. & Kalita J.K. (2011) Surveying port scans and their detection methodologies. The Computer Journal 54, pp. 1565–1581.

[40] Alliance N. (2016) 5g security recommendations package. White paper .

[41] Singh K., Singh P. & Kumar K. (2017) Application layer http-get flood ddos attacks: Research landscape and challenges. Computers & security 65, pp. 344–372.

[42] Calvert C.L. & Khoshgoftaar T.M. (2019) Impact of class distribution on the detection of slow http dos attacks using big data. Journal of Big Data 6, pp. 1–18.

[43] Xu C., Shen J. & Du X. (2021) Low-rate dos attack detection method based on hybrid deep neural networks. Journal of Information Security and Applications 60, pp. 102879.

[44] Kekki S., Featherstone W., Fang Y., Kuure P., Li A., Ranjan A., Purkayastha D., Jiangping F., Frydman D., Verin G. et al. (2018) Mec in 5g networks. ETSI white paper 28, pp. 1–28.

[45] 5GTN.

[46] Agiwal M., Kwon H., Park S. & Jin H. (2021) A survey on 4g-5g dual connectivity: road to 5g implementation. IEEE Access 9, pp. 16193–16210.

[47] Sperotto A., Schaffrath G., Sadre R., Morariu C., Pras A. & Stiller B. (2010) An overview of ip flow-based intrusion detection. IEEE communications surveys & tutorials 12, pp. 343–356.

[48] Sourdis I., Dimopoulos V., Pnevmatikatos D. & Vassiliadis S. (2006) Packet pre-filtering for network intrusion detection. In: *2006 Symposium on Architecture For Networking And Communications Systems*, IEEE, pp. 183–192.

[49] Li B., Springer J., Bebis G. & Gunes M.H. (2013) A survey of network flow applications. Journal of Network and Computer Applications 36, pp. 567–581.

[50] Potdar K., Pardawala T.S. & Pai C.D. (2017) A comparative study of categorical variable encoding techniques for neural network classifiers. International journal of computer applications 175, pp. 7–9.

[51] Client A. (2019) Ubuntu manpage: Ra - read argus(8) data. .

[52] Dissanayake M.B. (2021) Feature engineering for cyber-attack detection in internet of things .

[53] Guyon I. & Elisseeff A. (2003) An introduction to variable and feature selection. Journal of machine learning research 3, pp. 1157–1182.

[54] Cai J., Luo J., Wang S. & Yang S. (2018) Feature selection in machine learning: A new perspective. Neurocomputing 300, pp. 70–79.

[55] Stańczyk U. (2015) Feature evaluation by filter, wrapper, and embedded approaches. In: *Feature Selection for Data and Pattern Recognition*, Springer, pp. 29–44.

[56] Nasir I.M., Khan M.A., Yasmin M., Shah J.H., Gabryel M., Scherer R. & Damaševičius R. (2020) Pearson correlation-based feature selection for document classification using balanced training. Sensors 20, pp. 6793.

[57] Singh D. & Singh B. (2020) Investigating the impact of data normalization on classification performance. Applied Soft Computing 97, pp. 105524.

[58] Hossin M. & Sulaiman M.N. (2015) A review on evaluation metrics for data classification evaluations. International journal of data mining & knowledge management process 5, pp. 1.

[59] Grandini M., Bagli E. & Visani G. (2020) Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756 .

[60] Narkhede S. (2018) Understanding auc-roc curve. Towards Data Science 26, pp. 220–227.

9 LIST OF APPENDICES

Appendix 1 Tables related to feature selection
Appendix 2 Code for feature selection
Appendix 3 Code for ML model evaluations

Appendix 1 Tables related to feature selection

Table 9.1: Ranking from PCC for binary classification

Feature	PCC
sMeanPktSz	-0.586275
dTtl	-0.562927
CON	-0.530657
SrcPkts	-0.409042
sTtl	-0.402824
SrcBytes	-0.382140
TotBytes	-0.313687
TotPkts	-0.313170
dMeanPktSz	-0.309784
dHops	-0.277891
udp	-0.171260
e r	-0.166214
e i	-0.132709
sHops	-0.121578
SrcWin	-0.115640
Dur	-0.075601
e g	-0.037960
RSP	-0.035984
URP	-0.032955
FIN	-0.031316
e &	-0.028903
IdleTime	-0.028158
SrcTCPBase	-0.023536
SrcLoad	-0.017765
CLO	-0.011194
Rate	-0.009414
DstRate	-0.005708
e	-0.004386
DstTCPBase	-0.002806
e s	0.000260
URH	0.000936
TST	0.001039
PAR	0.001601
Load	0.002067
ECR	0.002294
SRC	0.004385
e *	0.004907
nan	0.008086
DstLoss	0.010298
SynAck	0.011245
e d	0.013665
AckDat	0.023935
TcpRtt	0.024803

Loss	0.039236
DstWin	0.041482
SrcLoss	0.044073
ACC	0.048298
Offset	0.049753
RST	0.053286
e s	0.054293
tcp	0.070020
INT	0.079350
icmp	0.095541
REQ	0.105914

Table 9.2: Ranking from PCC for multi class classification

Feature	PCC
udp	-0.639528
INT	-0.379360
sHops	-0.270401
dTtl	-0.254632
IdleTime	-0.198134
SrcBytes	-0.170545
dHops	-0.149132
SrcPkts	-0.143269
TotBytes	-0.131801
REQ	-0.121778
sMeanPktSz	-0.107542
TotPkts	-0.100205
sTtl	-0.090589
e r	-0.076001
icmp	-0.072593
SrcWin	-0.060693
e i	-0.058434
Offset	-0.052609
RSP	-0.016510
URP	-0.015120
e g	-0.014919
e &	-0.013261
TST	-0.008047
CLO	-0.005136
nan	-0.003252
DstTCPBase	-0.002969
SRC	-0.001764
ECR	-0.000922
e	-0.000822
PAR	-0.000644
e s	0.000849
URH	0.007558
DstRate	0.019422
SrcLoad	0.019892

Rate	0.021532
Load	0.024665
Dur	0.040427
e *	0.062180
CON	0.070229
SynAck	0.084669
SrcTCPBase	0.095418
DstLoss	0.115615
e d	0.127613
DstWin	0.137342
ACC	0.160435
dMeanPktSz	0.166248
FIN	0.213388
TcpRtt	0.222615
AckDat	0.231549
SrcLoss	0.235020
Loss	0.241873
e s	0.245781
RST	0.471204
tcp	0.672815

Table 9.3: Ranking from Chi-squared test for binary classification

Feature	Chi-square score
SrcTCPBase	1.419429e+11
SrcLoad	1.091107e+11
TotBytes	5.827505e+10
Load	4.387128e+10
SrcBytes	3.471654e+10
Offset	5.756779e+09
DstTCPBase	1.136704e+09
SrcWin	1.856636e+08
Rate	1.336199e+08
DstWin	7.574838e+07
sMeanPktSz	5.980309e+07
dMeanPktSz	5.685852e+07
DstRate	2.827583e+07
TotPkts	2.817141e+07
SrcPkts	1.350048e+07
CON	2.216865e+05
dTtl	1.959467e+05
sTtl	9.686312e+04
sHops	7.150701e+04
e r	2.369586e+04
udp	1.741522e+04
dHops	1.597209e+04
e i	1.465829e+04
Dur	1.303417e+04
REQ	7.012572e+03

icmp	5.274221e+03
INT	4.608248e+03
tcp	2.596853e+03
e s	2.230451e+03
RST	2.151872e+03
Loss	2.046727e+03
ACC	1.882514e+03
SrcLoss	1.848814e+03
RSP	1.239898e+03
e g	1.196791e+03
URP	1.115909e+03
FIN	8.402003e+02
e &	6.943431e+02
DstLoss	3.428742e+02
TcpRtt	1.644629e+02
e d	1.638304e+02
CLO	1.487878e+02
AckDat	1.485642e+02
IdleTime	1.038015e+02
nan	5.443994e+01
SynAck	3.206509e+01
e	3.174407e+01
e *	2.255999e+01
SRC	1.604970e+01
ECR	4.798780e+00
PAR	2.096946e+00
TST	6.855400e-01
URH	6.855400e-01
e s	4.032588e-02

Table 9.4: Ranking from Chi-squared test for multi class classification

Feature	Score
SrcTCPBase	6.538323e+13
Load	2.659567e+13
SrcLoad	5.437457e+11
Offset	5.290304e+11
TotBytes	5.856239e+10
SrcBytes	3.472641e+10
DstWin	8.439972e+09
DstTCPBase	8.048695e+09
Rate	3.613807e+09
DstRate	2.266833e+09
SrcWin	3.292015e+08
dMeanPktSz	2.268209e+08
sMeanPktSz	8.228229e+07
TotPkts	2.938248e+07
SrcPkts	1.388913e+07
sHops	2.412293e+06

icmp	5.719242e+05
Dur	5.510445e+05
udp	5.488008e+05
tcp	4.936795e+05
SrcLoss	4.888421e+05
RST	4.565805e+05
e s	4.437868e+05
Loss	4.299195e+05
CON	4.261361e+05
INT	3.220344e+05
REQ	2.670130e+05
ACC	2.039646e+05
dTtl	1.959693e+05
sTtl	1.543426e+05
FIN	9.642625e+04
DstLoss	8.284059e+04
AckDat	4.387975e+04
TcpRtt	2.938564e+04
e d	2.739721e+04
IdleTime	2.716303e+04
e r	2.369615e+04
SynAck	2.081327e+04
dHops	1.624021e+04
e i	1.468725e+04
e *	8.355237e+03
nan	2.956808e+03
RSP	1.239898e+03
e g	1.222403e+03
URP	1.115909e+03
SRC	8.717109e+02
e &	6.943431e+02
TST	4.308219e+02
ECR	2.606372e+02
CLO	1.487878e+02
PAR	1.138919e+02
URH	8.935807e+01
e	3.434522e+01
e s	4.456926e+00

Appendix 2 Code for feature selection

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso

df = pd.read_csv (r'/Users/sehan/Documents/Dataset/encoded2.csv')
display(df)

df = df.drop(columns=['Attack Type', 'Attack type based on tool'])
df = df.fillna(df.median())
display(df)

X = df.drop("Label",1)   #Feature Matrix
y = df["Label"]          #Target Variable

#test and train split (30% for test)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.30, random_state=0)

train_data = pd.concat([X_train, y_train], axis=1)
train_data
test_data = pd.concat([X_test, y_test], axis=1)
test_data

#Using Pearson Correlation
plt.figure(figsize=(45, 34))
cor = train_data.corr(method='pearson')
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()

def correlation(dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if  abs(corr_matrix.iloc[i,j]) > threshold:
```

Figure 9.1: Code for feature selection - Part 1

```
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr

corr_features = correlation(train_data, 0.90)
len(set(corr_features))

pandas.set_option('display.max_rows', None)
top = train_data.corrwith(train_data["Label"])
top = top.drop([' e        ', 'DstBytes', 'DstLoad', 'DstPkts', 'ECO',
'Max', 'Mean', 'Min', 'RunTime', 'SrcRate', 'Sum', 'pLoss'], axis=0)

top = top.drop(['Label'], axis=0)
top_abs = top.abs()
display(top_abs)
top_abs = top_abs.to_frame('Absolute PCC')
top_abs.index.name = 'Features'
top_sorted = top.sort_values()

df = df.drop(columns=[' e        ', 'DstBytes', 'DstLoad', 'DstPkts',
'ECO', 'Max', 'Mean', 'Min','RunTime', 'SrcRate', 'Sum', 'pLoss'])
X = df.drop("Label",1)   #Feature Matrix
y = df["Label"]

selector = SelectKBest(chi2, k=10).fit(X,y)
selected_features_df = pd.DataFrame({'Features':list(X.columns),
'Scores':selector.scores_})
selected_features_df.sort_values(by='Scores', ascending=False)

selected_features_df.columns
selected_features_df.Features.dtype

#selected_features_df = selected_features_df.drop('Features', axis=1)
df_norm = (selected_features_df.Scores-selected_features_df.Scores.min())/
(selected_features_df.Scores.max()-selected_features_df.Scores.min())
df_norm = pd.concat((selected_features_df.Features, df_norm), 1)

print("Scaled Dataset Using Pandas")
display(df_norm)

a, b = 0, 1
x, y = selected_features_df.Scores.min(), selected_features_df.Scores.max()
selected_features_df['normal'] = (selected_features_df.Scores - x) /
(y - x) * (b - a) + a
print (selected_features_df)
new = pd.merge(top_abs, df_norm, on='Features')
display(new)

new['Total'] = new['Absolute PCC'] + new['Scores']
```

Figure 9.2: Code for feature selection - Part 2

```
display(new)

new.sort_values('Total', ascending=False)
```

Figure 9.3: Code for feature selection - Part 3

Appendix 3 Code for ML model evaluations

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv(r'/Users/sehan/Documents/Dataset/Binary_New/
top_15_binary.csv')
df.head()

df = df.fillna(df.median())
print(df)

X = df.iloc[:,0:14].values
y = df.iloc[:,15].values

#Test and Train split (30:70)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.30, random_state=0)

#Data normalization - Z-score
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

#Data normaliztion - Min-max scaler
from sklearn.preprocessing import MinMaxScaler
scaling = MinMaxScaler(feature_range=(-1,1)).fit(X_train)
X_train = scaling.transform(X_train)
X_test = scaling.transform(X_test)

#Decision Tree

import time
from sklearn import tree
model = tree.DecisionTreeClassifier()
start = time.time()
model.fit(X_train, y_train)
stop = time.time()
print(f"Training time: {stop - start}s")

#Random Forest

import time
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=10,random_state=0)
```

Figure 9.4: Code for ML model evaluation - Part 1

```
start = time.time()
classifier.fit(X_train, y_train)
stop = time.time()
print(f"Training time: {stop - start}s")

#KNN

import time
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
start = time.time()
classifier.fit(X_train, y_train)
stop = time.time()
print(f"Training time: {stop - start}s")

#SVM

from sklearn.svm import SVC
model = SVC(kernel='linear', gamma='auto')
import time
start = time.time()
model.fit(X_train, y_train)
stop = time.time()
print(f"Training time: {stop - start}s")

#MLP

import time
from sklearn.neural_network import MLPClassifier
MLP = MLPClassifier(hidden_layer_sizes=(10,20,10), max_iter=1000)
start = time.time()
MLP.fit(X_train, y_train)
stop = time.time()
print(f"Training time: {stop - start}s")

#Prediction
start = time.time()
y_pred = model.predict(X_test)
stop = time.time()
y_pred
print(f"Prediction time: {stop - start}s")

from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, digits=4))
print(accuracy_score(y_test, y_pred))

#ROC plots
```

Figure 9.5: Code for ML model evaluation - Part 2

```
from sklearn.metrics import roc_curve, auc

model_fpr, model_tpr, threshold = roc_curve(y_test, y_pred[:,1])
auc_model = auc(model_fpr, model_tpr)

plt.figure(figsize=(5, 5), dpi=100)
plt.plot(model_fpr, model_tpr, linestyle='-', label='DT (auc = %0.8f)'
% auc_model)
plt.plot([0, 1], [0, 1], color="darkorange", lw=2, linestyle="--")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.legend()
plt.show()
```

Figure 9.6: Code for ML model evaluation - Part 3