REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Information Technologies

NETWORK AND SERVER INTRUSION DETECTION AUGMENTED WITH HYBRID DETECTION ENHANCEMENTS THROUGH DEEP LEARNING

Danya Ahmed Shehab ALALWAN

Master's Thesis

Supervisor

Asst. Prof. Dr. Ayça KURNAZ TÜRKBEN

İstanbul, 2024

NETWORK AND SERVER INTRUSION DETECTION AUGMENTED WITH HYBRID DETECTION ENHANCEMENTS THROUGH DEEP LEARNING

Danya Ahmed Shehab ALALWAN

Information Technologies

Master's Thesis

ALTINBAŞ UNIVERSITY

2024

The thesis/dissertation titled NETWORK AND SERVER INTRUSION DETECTION AUGMENTED WITH HYBRID DETECTION ENHANCEMENTS THROUGH DEEP LEARNING prepared by DANYA AHMED SHEHAB ALALWAN and submitted on …… (DATE) has been **accepted unanimously/by majority of votes** for the degree of Master of Arts/Master of Science/PhD in ……

Academic Title – First and Last Name of the Co-Supervisor	Academic Title – First and Last Name of the Supervisor

Thesis Defense Committee Members:

Academic Title - First/Last Name	Department, University	_____
Academic Title - First/Last Name	Department, University	_____
Academic Title - First/Last Name	Department, University	_____
Academic Title - First/Last Name	Department, University	_____
Academic Title - First/Last Name	Department, University	_____

I hereby declare that this thesis/dissertation meets all format and submission requirements of a …… (Master's/PhD) thesis/dissertation.

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Danya Ahmed Shehab AL ALWAN

Signature

DEDICATION

I dedication my thesis to:

University of Altınbaş

My very respected supervisor

All my family members, especially my very dear mother and father.

ABSTRACT

NETWORK AND SERVER INTRUSION DETECTION AUGMENTED WITH HYBRID DETECTION ENHANCEMENTS THROUGH DEEP LEARNING

ALALWAN, Danya Ahmed Shehab

M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst. Prof. Dr. Ayça KURNAZ TÜRKBEN

Date: 07/2024

Pages: 67

Network security and the identification of possible threats depend heavily on anomaly detection in network traffic. In order to determine which machine learning models are most useful in detecting network traffic anomalies, this research compared and evaluated a number of different machine learning models. Deep Learning with 10 epochs, Logistic Regression, Random Forest with a Filter method, Random Forest with a Wrapper method, and Random Forest with a Wrapper method were the models evaluated. A number of metrics were used for evaluation, including cross-validation, accuracy, F1-Score, ROC (Receiver Operating Characteristic), and precision- recall. The results showed that the Random Forest with the Wrapper approach and the Deep Learning model outperformed other assessment criteria. With a ROC score of 0.9716, the Deep Learning model had the best performance, clearly demonstrating its superior ability to differentiate between regular and abnormal network traffic. It also displayed outstanding precision-recall scores of 0.9621, indicating accurate anomaly identification with a low incidence of false positives. The Deep Learning model also attained a strong F1-Score of 0.8405, which represents a balanced mix of recall and precision.

Keywords: Cybersecurity, Intrusion Detection, Intrusion Detection System, Machine Learning, NIDS.

ÖZET

DERİN ÖĞRENME ARACILIĞIYLA HİBRİT TESPİT GELİŞTİRMELERİYLE AĞ VE SUNUCU GİRİŞİM TESPİTİ

ALALWAN, Danya Ahmed Shehab

Yüksek Lisans, Bilişim Teknolojileri, Altınbaş Üniversitesi,

Danışman: Dr. Öğr. Üyesi Ayça KURNAZ TÜRKBEN

Tarih: 07/2024

Sayfa: 67

Ağ güvenliği ve olası tehditlerin belirlenmesi, ağ trafiğindeki anormallik tespitine büyük ölçüde bağlıdır. Ağ trafiği anormalliklerini tespit etmede hangi makine öğrenimi modellerinin en yararlı olduğunu belirlemek için bu araştırma, bir dizi farklı makine öğrenimi modelini karşılaştırdı ve değerlendirdi. 10 dönemli Derin Öğrenme, Lojistik Regresyon, Filtre yöntemiyle Rastgele Orman, Sarmalayıcı yöntemiyle Rastgele Orman ve Sarmalayıcı yöntemiyle Rastgele Orman değerlendirilen modellerdi. Değerlendirme için çapraz doğrulama, doğruluk, F1 Puanı, ROC (Alıcı İşletim Karakteristiği) ve kesinlik-geri çağırma dahil olmak üzere bir dizi ölçüm kullanıldı. Sonuçlar, Wrapper yaklaşımına sahip Random Forest ve Derin Öğrenme modelinin diğer değerlendirme kriterlerinden daha iyi performans gösterdiğini gösterdi. 0,9716'lık bir ROC puanı ile Derin Öğrenme modeli en iyi performansı gösterdi ve düzenli ve anormal ağ trafiği arasında ayrım yapma konusundaki üstün yeteneğini açıkça gösterdi. Ayrıca, düşük yanlış pozitif oranıyla doğru anomali tanımlamasını gösteren 0,9621'lik olağanüstü kesinlik-geri çağırma puanları da gösterdi. Derin Öğrenme modeli ayrıca geri çağırma ve kesinliğin dengeli bir karışımını temsil eden 0,8405'lik güçlü bir F1 Puanı elde etti.

Anahtar kelimeler: Siber Güvenlik, Saldırı Tespiti, Saldırı Tespit Sistemi, Makine Öğrenimi, NIDS.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

HTTP	:	Hypertext Transfer Protocol
FTP	:	File Transfer Protocol
SMTP	:	Simple Mail Transfer Protocol
DNS	:	Domain Name System
DDOS	:	Distributed Denial of Service
PCA	:	Principal Component Analysis
SVM	:	Support Vector Machines
CNN	:	Convolutional Neural Networks

1. INTRODUCTION

1.1 GENERAL CONTEXT

The popularity and the rapid advancement of the Internet has made network security a growing problem. [1] More than ever are companies being compromised by malicious hackers and forced to pay a ransom. Hackers are also using more sophisticated techniques over the past few years, techniques that make intrusions harder to detect. To resist these attacks, cybersecurity experts have to develop more sophisticated techniques to mitigate these attacks. However, no method is perfect, and hackers will always be able to compromise the security system in some way. For that reason, security is a never-ending process of developing a security system, finding a way to compromise the security system and then again, patching the vulnerabilities in the security system. The introduction discusses security techniques of a specific category of security systems, namely Network Intrusion Detection Systems or NIDS; since new AI techniques for NIDS has shown its potential and because these techniques are very new, research in this field has not been completed.

1.2 ECURITY SYSTEMS

Security systems help to defend the infrastructure for potential attacks. There exist three main different security solutions to protect infrastructure: Firewalls, Antivirus software and Intrusion Detection Systems or IDSs. Each technique has its specific role and place in the infrastructure.

1.2.1 Firewalls

Firewalls serve as guardians at a computer's entry points or ports, regulating the flow of data exchanged with external devices. For instance, a rule might dictate, "Allow source address 172.18.1.1 to communicate with destination 172.18.2.1 via port 22." Visualizing IP addresses as houses and port numbers as rooms within them, the analogy suggests that only trustworthy individuals (source addresses) are granted entry into the residence (destination address). Once inside, access to specific rooms (destination ports) is further controlled, mirroring how different individuals within the house, like the owner, children, or visitors,

are restricted to certain areas. Owners have unrestricted access to all rooms (any port), while children and visitors are limited to designated spaces (particular ports), thereby ensuring security and privacy within the network environment [3][2].

1.2.2 Antivirus Software

Antivirus software is specifically engineered to scrutinize various forms of data, including web pages, files, software, and applications, with the aim of swiftly identifying and eradicating malware. Moreover, the majority of antivirus solutions on the market provide real-time protection, shielding devices from incoming threats by conducting routine system scans for recognized hazards and delivering automated updates. These programs are adept at detecting, thwarting, and eliminating harmful code and software. Their functioning involves cross-referencing programs and files against an extensive database containing known types of malware. Given the perpetual emergence of new viruses orchestrated by cybercriminals, antivirus software also remains vigilant in assessing computers for potential new or previously undiscovered forms of malware threats [4].

1.2.3 Intrusion Detection Systems

Intrusion Detection Systems (IDS) are an important instrument in current network security since they identify harmful incursions within a network (Figure 1.1). They are often used to monitor traffic; they can identify traffic abnormalities that suggest a malicious attack is occurring. IDS can be network-based, meaning it is placed in a network to scan for and identify threats by scanning all traffic. Host-based intrusion detection focuses on identifying intrusions through specific network endpoints. Two primary types of intrusion detection systems exist based on their intrusion identification methods: signature-based and anomaly-based. Signature captures the pattern of a group of packets which is defined in terms of their shared features. Attack signatures capture the patterns of attack packets. Signature-based intrusion detection is done by matching attack signatures. The matching process involves comparing the features of a new traffic packet with those presented in the existing set of attack signatures. On detecting the matched signature an alert is sent to the system administrator regarding the detected intrusion. The advantage of a signature-based detection approach is that it can detect known attacks with high accuracy and minimal false-positive

rate. This advantage follows a major drawback also; the signature-based approach is applicable to identify known attacks based on their signatures. It cannot detect unknown attacks for which signatures are not available in its knowledge base. 6 When a new attack is observed, the IDS developers will wait until a sufficient number of new attack packets are collected to create its signature and accordingly update the IDS knowledge base. The IDS which implements the signature-based detection approach is referred to as signature-based IDS. In Anomaly-based intrusion detection system the normal network behavior/activity is modeled in terms of the shared features of genuine traffic packets like the number of ports, the number of devices connected, bandwidth, protocols defined, etc. and detection is done by observing the deviations of a packet activity from the observed normal behavior model. Specifically, any packet that deviates from the baseline of normal behavior is suspected as an attack packet.

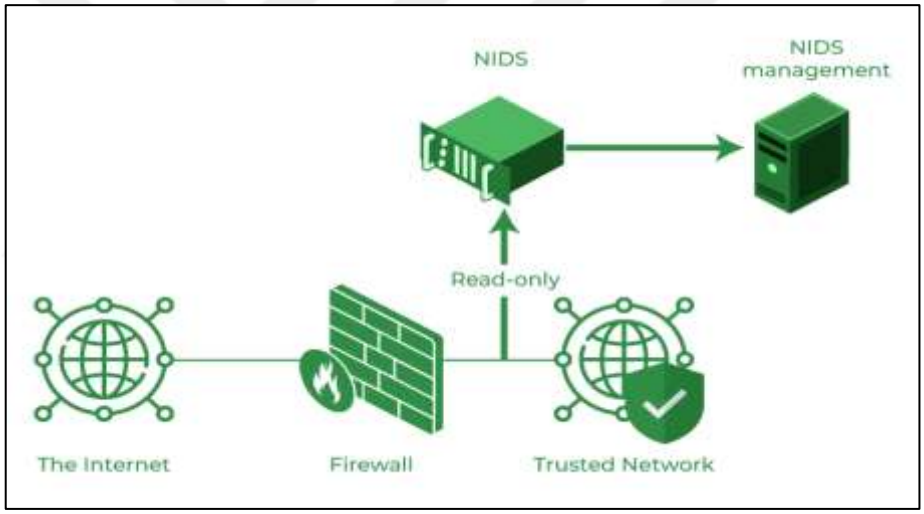

Figure 1.1: Intrusion Detection System.

Though anomaly-based intrusion detection systems are deployed to detect unknown attacks they run into a bottleneck on sensitivity control. Less sensitive anomaly-based IDSs may miss some critical intrusions whereas sensitive IDSs may raise too many false alarms, as a result of their higher False Positive Rate (FPR) control setting. Any hurtful action or infringement is normally detailed or accumulated halfway through a security data and occasion the board framework. A few IDS can answer an identified interruption when it is found. These are classed as Intrusion Prevention Systems (IPS), as seen in Figure 1.2 [5].

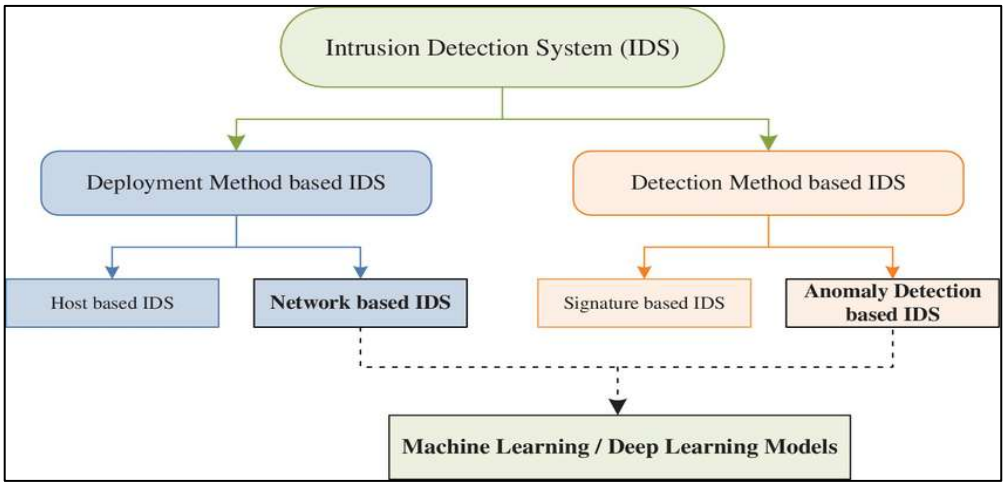

Figure 1.2: Intrusion Detection System Classification Taxonomy.

1.3 INTRUSION DETECTION SYSTEMS IN DETAIL

There are two types of intrusion detection systems: host-based and network-based. A Host Based Intrusion Detection System (HIDS) monitors a single host or device and alerts the user whenever suspicious behavior is identified (system files being updated or deleted, a suspicious system call, etc.). A Network Intrusion Detection System, or NIDS, is often installed at a strategic position, such as a gateway or router, and monitors for intrusion attempts in real time. IDS may be implemented in three distinct ways: misuse detection, anomaly detection, and hybrid detection [6].

1.3.1 Misuse Detection

In misuse detection, the IDS uses a set of rules to identify attack types. Abuse recognition can be grouped into an information based framework and an AI framework. In an information based framework, network information is contrasted with predefined rules or assault designs, where AI based IDS attempts to find the assault classes without help from anyone else. An AI model attempts to recognize the assault classes. The purpose of a machine learning based IDS is to provide a general representation and distinction of the attack classes. This is where misuse detection fails. Misuse detection is not able to detect unknown attacks and they also need regular updates to detect newer attacks.

1.3.2 Anomaly Detection

Anomaly detection-based IDS tries to distinguish between normal and abnormal internet traffic. It is based on the hypothesis that the network traffic of the average user must be different from that of an attacker. Anomaly detection models the normal use of the system and updates this model over time. For example, each network connection is identified by a set of parameters, the behavior of those parameters is monitored over time, any abnormal deviation is marked as anomalous. Anomaly detection can be implemented with statistical data, machine learning and Finite State Machines or FSMs.

1.4 MOTIVATION

IDS is of great importance for its help to identify anomalies and prevent attacks and also for securing the system. Intrusion Detection challenges are receiving false alarms, accuracy detection in low rate, datasets are not balanced, and delay in getting response. The background scene of many captured intrusions is having irregular pattern which makes the intrusion detection process more complex. Also, many software industries require automatic IDS which motivates us to work on this research more effectively. The motivation for IDS researchers is to develop a system that can detect intrusions and these IDS are widely used for securing information or to safeguard the devices. Additionally, unsupervised learning techniques are being explored, which simulate typical network behavior and raise alerts upon detecting anomalies. This research focuses on employing a supervised approach, leveraging labeled network data for training purposes.

1.5 PROBLEM STATEMENT

This thesis investigates anomaly detection of web-based attacks on microservices applications using application performance indicators and service logs. The fundamental concept is to create a normal activity profile based on the (simulated) typical activity on the online application, and then detect anomalies such as web assaults as differences from the usual behavior. This work will be carried out by constructing a dataset solely comprising normal activity and then training machine learning models to discriminate between the taught behavior and other behaviors.

1.6 OBJECTIVES

The success of this project is dependent on our ability to develop a reliable machine learning-based system for detecting abnormalities in network data, which is critical for system security. As more apps use networks, effective security measures are more important than ever. Detecting anomalies is a crucial component of network security because it might disclose hidden vulnerabilities. We anticipate that by using machine learning techniques to spot anomalies in network data, administrators would be able to take more preemptive actions against security breaches. We will leverage open-source data sets such as NSL-KDD, CICIDS2017, and UNSW-NB15 to complete the task. The performance of machine learning algorithms can be enhanced by preprocessing the dataset to remove noise, missing values, and other anomalies. Normalization, transformation, and feature selection are examples of data preprocessing methods.

The project's fourth aim is to implement the most effective machine learning model for real-time anomaly detection. The system requires real-time network traffic anomaly identification and alerting to system administrators. The selected machine learning model will be integrated into a real-time network monitoring system. Anomalies, both known and unknown, will be highlighted so that network management can implement remedial actions.

1.7 CONTRIBUTION

This thesis endeavors to devise and implement a robust approach for identifying web-based attacks within a microservices framework. Presently, the majority of research on anomaly detection in web-based attacks primarily focuses on monolithic web applications, often overlooking the significant shift towards cloud computing. This study aims to fill this gap by specifically addressing the challenges posed by the evolving landscape of cloud-native architectures, where microservices play a pivotal role. Moreover, the material available doesn’t experiment on realistic web applications but on simple applications used as proof of concept, mostly considering network traffic as training data. The application used as our testing environment is quite complex (40+ microservices) and has been deployed on a real, on premise kubernetes cluster.

The design and evaluation of machine learning models will be performed in an unsupervised manner, which is another point of contribution. This choice has been made to reflect the lack of labeled data in real intrusion detection scenarios, often having to deal with unlabelled or partially labelled data. The approach consists in combining two sources of information:

i. Performance metrics: CPU utilization, memory usage, network volume etc...

ii. Application logs: messages produced by every application pod regarding its behaviour. and combining them to try to detect cybersecurity attacks targeting the application.

The contributions can be summarized as follows:

i. Deployment of a complex microservice application.

ii. Usage of log production as added features.

iii. Design of an unsupervised approach to the problem

1.8 SCOPE AND LIMITATION

The goal of this study is to create a network traffic anomaly detection system based on machine learning. The following elements are covered within the study's range of inquiry.:

Analysis of network traffic data: The study's main focus will be on the analysis and comprehension of network traffic data, particularly the elements and traits important for anomaly identification.

Investigation of machine learning algorithms: Anomaly detection in network traffic will be a focus of the research as it examines several supervised machine learning algorithms.

Model development and optimization: The development, validation, and improvement of machine learning models for spotting anomalies in network data will all be part of the project. To achieve optimum performance, the models will be adjusted utilizing hyperparameter optimization methods.

Implementation and evaluation of the anomaly detection system: The study's implementation of the selected machine learning model in a real-time network monitoring system and performance assessment of it using acceptable metrics will be covered.

However, there are limitations to the study, which include:

Dataset limitations: The study makes use of freely accessible datasets like NSL-KDD, CICIDS2017, and UNSW-NB15. The ability to generalize the results to other network traffic datasets may depend on how well- representative and high-quality these datasets are.

Algorithmic limitations: The study will concentrate on a particular set of machine learning algorithms, which may not be comprehensive or indicative of all potential methods for anomaly identification in network traffic.

1.9 THESIS OUTLINE

The second chapter of this thesis paper discusses related studies on current intrusion detecting algorithms. The implementation approach and network training options are covered in Chapter 3. Chapter 4 elaborates on the experiments and their outcomes. The experimental results are thoroughly explored. Chapter 5 presents the conclusions and recommendations for further study.

1.10 CONCLUSION

In order to improve network security and defend against cyber threats, this research project created a machine learning-based anomaly detection system for network traffic data. The Random Forest algorithm was the focus of the study's investigation of different machine learning algorithms for the building, improvement, and testing of models. Data pre-processing, feature extraction, and model validation were all part of the system's analysis and design, while data processing, model building, and testing were all part of the system's implementation. However, the present IDS is affected by its occurrence of alarm in wrong time, low identification rate, datasets are unbalanced and delayed time in response which leads to misclassification of intrusions in various scenarios. Hence, there is a requirement

for developing an automated IDS that works well in different scenarios when compared to the existing system.

2. LITERATURE REVIEW

2.1 INTRODUCTION

A network-based Intrusion Detection System, sometimes known as a NIDS, is a security hardware that is strategically placed to monitor vital real-time network traffic. This research study is aimed to build a framework for IDS. It is critical in order to protect the software and product applications as well as their basic functionality. The research focuses on detecting intrusions in various sorts of network traffic attacks. The chapter concludes by highlighting the importance of evaluating machine learning models and by going into greater detail on important performance indicators and widely used benchmark datasets in this field. The goal of this literature review is to provide a comprehensive understanding of the current state of research in network traffic anomaly detection using machine learning, laying the groundwork for the development of a ground-breaking research framework that addresses the shortcomings and gaps found in the existing literature [10]. Developing intrusion detection systems involves various techniques associated with machine learning and transfer learning approaches. As part of this research work, a critical survey on various techniques of developing efficient intrusion detection systems was carried out. Hence, a comprehensive survey is presented in chronological order of the publications related to computational approaches to IDS.

2.2 NETWORK TRAFFIC

Network communication relies on a standardized set of protocols that facilitate data transmission across interconnected systems. Illustrated in Figure 2.1, the Internet Protocol Stack comprises five layers.

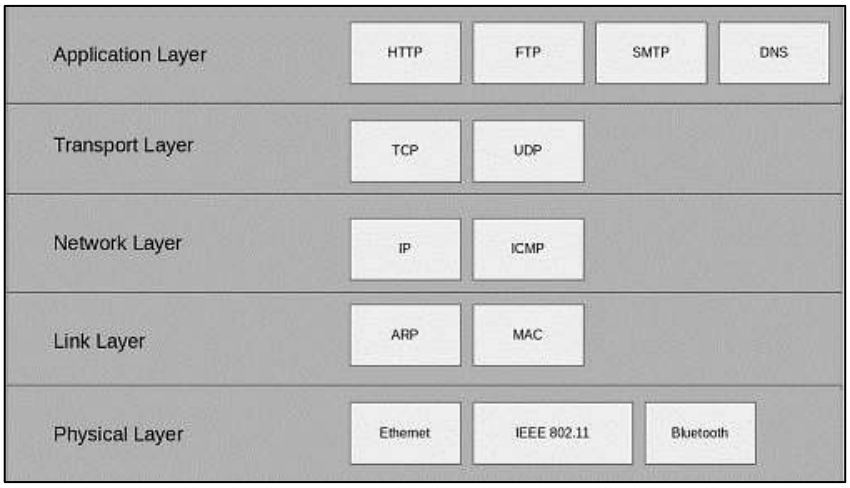

Figure 2.1: The Internet Protocol Stack with Examples for Each Layer.

With them, it is feasible to move information universally to other associated has. Practically speaking, nonetheless, these addresses are not exceptionally doled out: The real host that is recognized by a specific location relies upon the subnet. There are two variants of IP being used, the more seasoned IPv4 and the later IPv6 which is intended to supplant the previous, giving a greater arrangement of potential locations. The portions of an IPv4 bundle are displayed in table 2.1 [13]. Every one of the sections showed there, with the exception of the payload, have a place with the IP header.

Table 2.1: IP Packet Segments in IPv4.

Name	Length	Description
Version	4 bits	the version of the IP packet
Header Length	4 bits	the number of 32-bit segments in the packet's header
Type of Service (TOS)	8 bits	information about the type of the packet's data; used to distinguish real-time and non-real-time traffic
Datagram Length	16 bits	total length of the IP datagram
Identifiers	16 bits	used for identifying connected fragments of one IP packet
Flags	3 bits	control options that are used for fragmentation
Fragmentation Offset	13 bits	used for reassembling multiple fragments of an IP packet
Time-to-live	8 bits	a counter that gets decremented by one each time the packet passes a router in the network; to prevent endless circulation inside a network, the packet is dropped when this value reaches zero
Protocol	8 bits	uniquely identifies the transport-layer protocol of the IP packet's payload (e.g.: 6 for TCP and 17 for UDP)
Header Checksum	16 bit	computed by using 1s complement arithmetic; helps to detect bit errors in the header
Source IP address	32 bit	IP address of the sender
Destination IP address	32 bit	IP address of the recipient
Options	variable	placeholder for rarely-used options
Data / Payload	variable	the transport-layer data that is to be delivered by the IP packet

The Client Datagram Convention works by utilizing IP parcels to send information in a connectionless and temperamental way, demonstrating no confirmation of message conveyance. By and by, it gives simple uprightness checks to sent information and incorporates source and objective ports, working with the relationship of a message with an interaction on the particular host. With a simple 8-byte header affixed to the parcel, UDP finds normal utilization in sight and sound streaming, web correspondence, and directing conventions. On the other hand, the Transmission Control Convention (TCP) is as often as possible picked as a trustworthy vehicle layer convention. Rather than UDP, TCP guarantees message precision through blunder discovery, retransmissions, and header fields for succession and affirmation numbers. Additionally, it operates in a connection-oriented manner, establishing a connection between hosts through a three-way handshake before actual message transmission, wherein specific header bytes ("flags") are exchanged between the communicating hosts. In a more detailed explanation, the process begins with the

initiating host initiating the connection by transmitting a packet with the SYN flag set. The designated host then responds with a packet containing the ACK flag, signifying acknowledgment of receipt. This connection establishment process may conclude with the FIN flag, signifying closure, or it can be terminated abruptly using the RST flag.

2.3 NETWORK FLOWS

It is frequently alluring to have understanding into the qualities of an organization, for example for guaranteeing its well-working, for portraying its usage or for identifying vindictive way of behaving. To accomplish this, the organization traffic can be checked. There are two principal systems for doing as such: (1) catching the crude bundles or (2) the distinguishing proof of organization streams [15].

2.4 NETWORK VULNERABILITIES AND ATTACKS

Network weaknesses are "innate shortcomings in the plan, execution and the board of an organized framework" [16]. An organization assault is a "grouping of tasks that puts the security of an organization or PC framework in danger" [17], by taking advantage of a weakness. It ought to be noticed that it isn't generally imaginable to separate between incidental breakdown because of programming bugs or human wrongdoing on the one side, and purposeful organization assaults on the opposite side, as they could seem comparative on a specialized level (for example at the point when network bundles are noticed). Network assaults are typically led determined to think twice about framework's accessibility, secrecy, or honesty. They can generally be partitioned into inactive and dynamic assaults, in light of the way of behaving of the going after subject [18]. An endeavor at a more definite characterization is displayed in the following segment.

By grouping network assaults and characterizing a scientific classification, it is expected to track down a predictable way for determining connections between various assaults and for concluding which assaults share similitudes or are unique in relation to one another [19]. Notwithstanding, there are different organization assault scientific categorizations proposed in writing [20], and most goes after can't plainly be put exclusively into one classification,

which confounds the possibility of a various leveled organizing. The accompanying assault classifications are chosen from the scientific categorizations depicted by [21], [22] and [23].

Denial of Service (DoS) Attacks: looks to deliver an organization administration inaccessible for real clients. This pernicious objective can be achieved through different techniques, for example, purposely prompting the casualty's framework to come up short, frequently by taking advantage of weaknesses like cushion flood (as exemplified by the Ping of Death assault), or by debilitating its assets, as shown in a TCP SYN flood. Conversely, a Conveyed Forswearing of Administration (DDoS) assault happens when the attack is organized from numerous particular IP addresses at the same time. Unlike a traditional DoS attack, blocking a single IP address will not suffice to halt an ongoing DDoS assault, as it involves coordinated efforts from multiple sources [24] [26].

Infection assaults: These attacks endeavor to implant malicious software onto the target system, including worms, viruses, or trojans. Worms and viruses are both self-replicating programs capable of causing harm independently or facilitating further attacks by multiplying themselves [27] .

Information Gathering Attacks: These attacks aim to acquire information about a system or its data without causing immediate damage. For instance, one tactic involves scanning a host to identify open ports, commonly referred to as probing. Alternatively, passive attacks, like sniffing or eavesdropping, are employed to covertly gather information. By targeting the network layer, attackers can passively intercept data as it is transferred, compromising the confidentiality of the communication.

2.5 INTRUSION DETECTION SYSTEMS TAXONOMY

Intrusion Detection Systems (IDSs) are the hardware or software systems that constantly monitor the cyberspace for detecting the intrusions [17]. On detection of an intrusion, IDS generates an alert to the system administrator to follow-up further action in response to the alert. Intrusion detection systems are deployed next to the firewalls in the detection process and their main motto is defense-in-depth. Firewalls don't go through the internal details of the traffic, whereas, intrusion detection systems employ detailed detection procedures by capturing the internal and implicit details like the signature of the traffic. IDS consists of

many components; monitoring system, data gathering devise, ID engine, knowledge base, configuration device, and response component. Data gathering devise receives raw data from the monitoring system and transforms it to identify events which will be submitted to the ID engine for detection of possible intrusions. ID engine analyses the submitted data by taking required information from the knowledge base and configuration devices.

2.5.1 Signature-Based IDS

Signature captures the pattern of a group of packets which is defined in terms of their shared features. Attack signatures capture the patterns of attack packets. Signature-based intrusion detection is done by matching attack signatures. The matching process involves comparing the features of a new traffic packet with those presented in the existing set of attack signatures. On detecting the matched signature an alert is sent to the system administrator regarding the detected intrusion. The advantage of a signature-based detection approach is that it can detect known attacks with high accuracy and minimal false-positive rate. This advantage follows a major drawback also; the signature-based approach is applicable to identify known attacks based on their signatures. It cannot detect unknown attacks for which signatures are not available in its knowledge base. When a new attack is observed, the IDS developers will wait until a sufficient number of new attack packets are collected to create its signature and accordingly update the IDS knowledge base. The IDS which implements the signature-based detection approach is referred to as signature-based IDS.

2.5.2 Anomaly-Based IDS

In Anomaly-based intrusion detection system the normal network behavior/activity is modeled in terms of the shared features of genuine traffic packets like the number of ports, the number of devices connected, bandwidth, protocols defined, etc. and detection is done by observing the deviations of a packet activity from the observed normal behavior model. Specifically, any packet that deviates from the baseline of normal behavior is suspected as an attack packet. The suspected anomaly could be either a known attack or an unknown attack, the details of which are given in the next sections of this chapter [31]. Though anomaly-based intrusion detection systems are deployed to detect unknown attacks they run into a bottleneck on sensitivity control. Less sensitive anomaly-based IDSs may miss some

critical intrusions whereas sensitive IDSs may raise too many false alarms, as a result of their higher False Positive Rate (FPR) control setting.

2.5.2.1 Statistical-based techniques

Each false alarm is expensive as it calls for security expert intervention to decide whether it is an attack or not. So, it is better to reduce false alarms while keeping the detection accuracy as high as possible in an intrusion detection system. [33].

2.5.2.2 Knowledge-based techniques

To combine the advantages of both signature-based and anomaly-based approaches, some of the researchers proposed hybrid systems [Tes15] where a hybrid IDS implements both signature-based and anomaly-based approaches. Hybrid IDS is implemented in two phases. Initially, all known attacks are detected using a signature-based detection approach in the first phase. Later, all the benign traffic filtered from the first phase is sent to the anomaly-based detection approach in the second phase, where suspicious unknown attack packets are separated from the benign packets. In other words, once the signature-based IDS filters out the known attack packets, the residual benign traffic when passed through the anomaly-based IDS gets separated into normal traffic and the suspicious unknown attack traffic. This combined advantage gives more robustness in detecting attacks, although perhaps at the cost of operating a bit slowly, possibly, causing a lag in detection [34].

2.5.2.3 Machine learning-based techniques

Intrusion detection systems need to analyze huge amounts of traffic information for extracting hidden patterns of features to form attack signatures and hence require Machine Learning (ML) techniques. Machine learning methods are broadly categorized as supervised and unsupervised methods. Supervised methods predict class labels of instances based on the knowledge learned from a large collection of labeled data. Whenever labeled data is not available unsupervised methods like clustering, statistical modeling, etc., are used as they do not require labeled data for pattern extraction. Since intrusion detection systems generally maintain labeled data related to known attacks and genuine traffic packets (benign), supervised methods are mostly preferred for the construction of signature-based IDS. Machine learning has proven its potential for developing effective IDS for detecting known

attacks using the signature-based approach. Many supervisory learning algorithms like K-Nearest Neighbor (KNN) [18], Decision Tree (DT) [17], Random Forest (RF) [16], Support Vector Machine (SVM) [16], Naïve Bayes (NB) [15], Logistic Regression (LR) [8], etc., are extensively used for implementing signature-based IDS, and a few of them are described below in brief .

2.6 MACHINE LEARNING

Supervised learning algorithms are trained using labeled data but Unsupervised learning algorithms are trained using unlabeled data. Supervised learning model takes direct feedback to check if it is predicting correct output or not but Unsupervised learning model does not take any feedback. Supervised learning model predicts the output but Unsupervised learning model finds the hidden patterns in data. Supervised learning addresses Classification and Regression based problems and Unsupervised Learning can be applied to Clustering and Association problems.

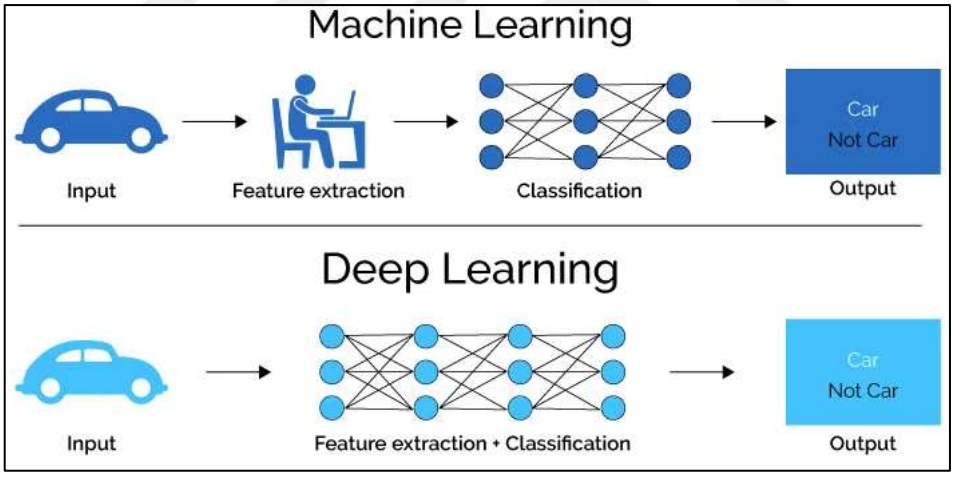

Figure 2.2: Machine Learning Diagram.

Clustering means a process of separating a class of extracted data into a group of related data. To initiate the clustering process, first partition the data set and group it based on its nature. Assigning the label for the group totally depends on its characteristics. A cluster is a group formed based on the attributes (data). The objects present in the same cluster are similar. The clustering-based technique helps to disclose the original structure of an interpreted data. The instance that isn't a member of any cluster is known as an outlier.

2.7 RELATED WORKS: NIDS BASED MACHINE LEARNING

Over the past years, deep learning algorithms have experienced rapid advancement, garnering significant attention within the field of artificial intelligence. Deep learning, depicted as a subset within the classification of machine learning algorithms in Figure 2.1, originates from the broader category of Artificial Neural Networks (ANN). Since its inception in 1943, when [40] first proposed the concept of a mathematical model inspired by biological neurons, ANN has undergone substantial refinement. [41] introduced weight adjustments, laying the groundwork for subsequent enhancements, while [42] constructed the first learnable ANN, the Perceptron, in 1958. In a pivotal development, [43] proposed the backpropagation algorithm (BP) in 1986, further refining and shaping ANN. Today, ANN stands as one of the most widely utilized machine learning algorithms, characterized by multiple layers of neurons interconnected through adaptive weights, forming what is known as a fully connected network. According to the universal approximation theorem in ANN's mathematical framework, a single hidden layer within the network can approximate any continuous function that maps a real interval to a real output interval, provided that the gradient of the activation function does not vanish. To put it differently, a single hidden layer ANN has the capability to represent any nonlinear continuous function, effectively embodying the essence of shallow learning. Shallow learning, exemplified by such single hidden layer ANNs, distinguishes itself by its ability to characterize complex nonlinear relationships. However, it's worth noting that feature selection in shallow learning is conducted independently, separate from the ANN architecture itself. This segregation of feature selection remains a common deficiency across various machine learning algorithms discussed earlier. By treating feature selection as a distinct step, rather than integrating it seamlessly within the ANN framework, these approaches may encounter limitations in fully harnessing the potential of the data and optimizing model performance.

In their review, [44] utilized a one-class support vector machine (SVM) to distinguish peculiarities inside the KDDCUP99 dataset, contrasting its presentation with that of a probabilistic brain organization and a regular two-class SVM. They detailed astoundingly high identification rates, almost arriving at 1, for Disavowal of Administration (DoS) and testing assaults, alongside reliably high precision and F1-score measurements for the one-

class SVM. In the interim, [45] led a comparative examination looking at one-class SVMs utilizing straight and Gaussian portions with two-class SVMs. They created an engineered dataset in light of both remote and Ethernet LAN traffic, enveloping five particular assault types, incorporating three HTTP assaults in remote organizations, one taking advantage of the IEEE 802.11 remote convention, and another containing port filtering assaults. Very, the features created for the Ethernet LAN network were not stream based; taking everything into account, they included estimations, for instance, current throughput and the number and assignment of open ports assessed one time each second. While good outcomes were accomplished for different assault types, testing assaults showed an identification pace of just 61.37%, joined by a low bogus positive pace of 1.15%. In their survey, [46] took on a specific strategy through setting up a one-class support vector machine (SVM) using toxic traffic isolated from a honeypot dataset. Using hyperparameter improvement strategies, they successfully refined a zero sham recognizable proof rate, contingent only upon components, for instance, source and goal ports, TCP flags, and the IP show. Nonetheless, a remarkable restriction emerges from the utilization of various datasets for extricating harmless and vindictive traffic, possibly presenting an inclination that may not reflect genuine situations. In addition, preparing the one-class SVM exclusively on vindictive traffic expects that future assaults during testing will look like those saw during preparing. Also, a few examinations have featured the failure of help vector machines, especially while managing huge and high-layered datasets.

In their study, [34] effectively employed Kernel Principal Component Analysis (KPCA) as part of their model, although the remainder of the model operated in a supervised manner. Meanwhile, [47] adopted a two-stage methodology, leveraging the two-dimensional encoding of a deep autoencoder to derive input features for a one-class SVM. Contrasting their methodology and a plain one-class SVM utilizing unreduced highlights, they revealed huge enhancements, with an accuracy of 99.97% and a review of 98.28%. In addition, they accomplished eminent decreases in both preparation and testing times. Be that as it may, the plain one-class SVM actually yielded good accuracy (96.26%) and review (98.21%) measurements, displaying its adequacy in spite of the upgraded presentation of the proposed approach.

Notwithstanding the previously mentioned approaches, a few different examinations influence autoencoders for highlight decrease in peculiarity location. For example, in a paper by [47], scanty autoencoders are utilized to separate highlights, which are then scholarly by a softmax relapse classifier. In spite of the fact that they noticed an expanded F-Measure esteem with autoencoders, their design just incorporated a solitary secret unit. An upgrade could include developing the organization's design and using unaided classifiers rather than relapse. Other exploration endeavors, for example, [49], propose the utilization of (profound) autoencoders for arrangement. In this model, just harmless cases are utilized for preparing, with the information split into preparing and approval sets. When defied with obscure traffic, the autoencoder recognizes assaults in view of an occurrence's reproduction blunder surpassing a foreordained limit. Deciding the ideal edge esteem and the autoencoder's engineering, including the quantity of layers and neurons per layer, are treated as not entirely set in stone through the irregular pursuit calculation. For the CIC-IDS-2017 dataset, the ideal boundaries bring about a profound autoencoder highlighting three secret layers, containing 18 hubs in both the info and result layers, 15 hubs in the first and third secret layers, and 9 hubs in the focal secret layer. In their work, [50] presented a group of autoencoders intended for circulated arrangement with negligible runtime necessities. They remove measurable elements from a constant stream of organization bundles utilizing damped steady measurements, hence planning these highlights to more modest subsets used as contributions for the outfit of autoencoders. Each autoencoder inside the gathering comprises of three layers and delivers a reproduction blunder for its relegated subset. The center of the peculiarity recognition module contains another autoencoder, which gets the totaled recreation mistakes from the group autoencoders as sources of info. This autoencoder produces a score demonstrative of the oddity level in the reviewed parcels. On the other hand, [51] proposed "AutoIDS," utilizing two parallelly prepared autoencoders explicitly prepared on typical traffic. The first autoencoder, a scanty variation, uses sparsity as an order model, while the second spotlights on the recreation mistake. During traffic classification, the sparse autoencoder is queried initially due to its faster processing time. Only when uncertain, the traffic is then presented to the second autoencoder, which, although slower, offers higher accuracy. Through this methodology, they achieved impressive precision (95.59%) and recall (97.43%) on the NSL-KDD dataset.

As of late, there has been a flood in using profound learning strategies for independently knowing highlights inside network traffic payloads. These techniques have witnessed significant advancements across various domains of computer science, prompting a growing research interest in their application to Network Intrusion Detection Systems (NIDS). According to a survey, 14 out of 26 recent papers have adopted deep learning methods [52]. One such approach, presented by [53], involves transforming raw network traffic data into two-dimensional images, subsequently utilized by a convolutional neural network (CNN) to extract features. CNNs, renowned for their efficacy in image classification, enable the extraction of spatial features from images. The paper talks about two particular methodologies: the first utilizes pictures produced from the whole organization stream as contribution for the CNN, while the second creates one picture for every parcel, catching fleeting relations among bundle vectors. [54] recognizes that their methodology misses the mark contrasted with contending strategies, with discovery paces of just 74.19% and 64.25%, separately. The authors emphasize the need for further refinement to address the challenges posed by imbalanced datasets and to incorporate traditional traffic features for enhanced performance. In contrast, [54] explores the application of Natural Language Processing (NLP) techniques to bolster NIDS performance. They use a text-convolutional brain organization (CNN) to separate elements from bundle payloads, utilizing a byte-level word implanting technique likened to word2vec. Not at all like its run of the mill utilization for saving semantic relations between words, here word2vec is adjusted to treat every byte in a stream as a word, connecting parcel payloads inside each organization stream. The subsequent embeddings are then utilized for highlight extraction utilizing a Text-CNN, close by measurable elements from bundle headers. Training a Random Forest algorithm with these features yields impressive classification accuracy (99.13%) and a low false positive rate (1.18%), outperforming other learning algorithms compared in the study. However, the supervised training of both the CNN and classification algorithm limits real-world applicability by hindering detection of unknown attacks. Moreover, the utilization of the ISCX2012 dataset, recorded in 2012 [55], may not adequately reflect contemporary attack types.

2.8 CONCLUSION

With a focus on machine learning methods in particular, this literature review has offered a thorough overview of the status of research in network traffic anomaly detection. It has discussed a range of topics related to network traffic analysis, such as deep packet inspection, flow analysis, protocol analysis, and packet capture and analysis, emphasizing the significance of comprehending network behaviour to identify potential security threats and enhance network performance.

In the review, numerous network abnormalities, including point, contextual, and collective anomalies, as well as how they materialize in various network security risks, such as DDoS attacks, port scanning, malware activities, and unauthorized access attempts, have been covered. The shortcomings of conventional anomaly detection techniques, such as rule-based methods, statistical techniques, and signature-based techniques, have been thoroughly investigated, highlighting the want for more flexible and reliable solutions.

Security in an interconnected computer world is a prime concern for casual users to sophisticated scientific and research users. The presence of huge user sources operating with multivariate intentions emphasizes the need for robust network security. The availability of tools to intrude networks has seen a rapid growth. The ease of access to Intruding tools signifies the fact that developing an efficient Network Security tools to counter intrusions is anticipated.

3. PROPOSED SYSTEM

The researchers of data science aim at getting actionable insights from raw data by applying techniques from multiple fields including statistics and machine learning. Machine learning provides many supervised learning algorithms like K-Nearest Neighbor (KNN), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Artificial Neural Network (ANN), Rule-based classifiers and Logistic Regression, etc. that support for building IDS models. The suitability of a model is determined based on the type of features. Specifically, ANN, Logistic Regression, KNN, etc. are preferred to build classifiers using numeric features while, DT, RF, Rule-based classifiers, etc. supports building classifiers by involving categorical features.

3.1 METHODOLOGY

The proposed framework of machine learning based IDS is given in Figure. 3.1. The collection of labeled data is partitioned into training, validation, and testing data. As shown in the figure, data is initially sent to the pre-processing module where appropriate feature engineering is done to prepare the data for building classification models. The classifier is learned using the pre-processed training data, and the classification model is refined by varying the optimal model parameters which are fixed by validating the model using the validation data. Finally, the generalization accuracy of the classifier is assessed using the test data, and if it is acceptable, the classifier will be used to make predictions on unknown data.

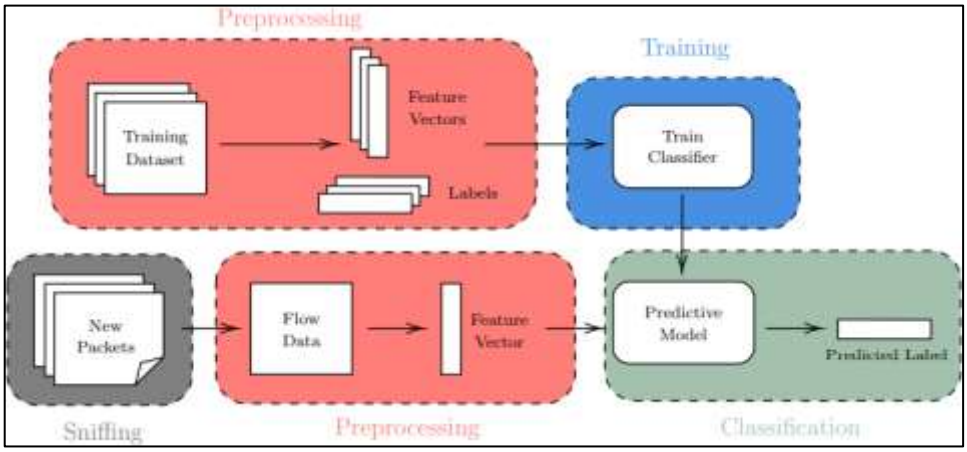

Figure 3.1: The IDS Pipeline.

In essence, the classifier learns attack patterns/signatures in the training phase using known labeled data and uses them for making predictions on the incoming network traffic data with an implicit assumption that both known labeled data and the new incoming network traffic data possess the same characteristics in terms of feature spaces and data distributions. Along with this stand-alone approach of using a single classifier for implementing intrusion detection systems, there exist two other approaches for implementing intrusion detection systems; one is the hybrid learning approach and the other is the ensemble approach.

3.1.1 Datasets

For this work, around 35 publications are carefully analyzed and an experiment is conducted on a Windows PC from HP with a dual-core processor and 8 GB of RAM. The workbench supplied by an Anaconda platform is called Jupyter, and it is the environment on which the application was developed. Python was utilized as the programming language to create this intrusion detection model. UNSW NB15, a dataset with a 68 percent training set and a 32 percent testing set that consists of 175,341 tuples in the train set and 82,332 tuples in the test set, was used for this study. We preprocessed the data first in order to use it to make wise conclusion Then we applied feature selection to pick high rank features. We have applied supervised machine learning algorithm to classify the data instances into the attacks and normal. In second phase, unsupervised machine learning algorithm is used to make the 9 clusters for nine types of attacks

3.1.2 Data Pre-processing

Data pre-processing should be done to prepare the data before building a classifier. Pre-processing is the task of cleaning and transforming the dataset and making it ready for model construction. Models constructed without proper pre-processing may not give reliable results Pre-processing generally includes the following tasks:

Handling missing values: The performance of machine learning algorithms can be negatively impacted by missing or incomplete values in the dataset. Depending on the type and quantity of the missing data, missing values were first recognized and either filled in using the proper techniques or eliminated. Mean or median imputation, nearest-neighbour imputation, or

utilizing a model-based approach to estimate the missing values are common techniques for filling in missing values.

Encoding categorical features: The majority of machine learning algorithms can only handle numerical representations of categorical information, such as protocol kinds or network service types. Label encoding assigns a unique integer value to each category in a categorical feature, while one-hot encoding represents each category with a binary vector where only one element is hot (1) indicating the presence of that category and all other elements are cold (0). The algorithm being utilised and if the categorical feature has an ordinal relationship between its values determine which of these approaches should be used.

Normalization of numerical features: Different units or scales for numerical characteristics can result in unequal contributions to the model and poor algorithm performance. Numerical features are scaled using methods like Min-Max scaling or standardisation to address this issue (also known as z- score normalization). While standardisation changes the characteristics to have a zero mean and unit variance, min-max scaling linearly adjusts the features to a given range (for example, [0, 1]). The algorithmic choice as well as the data distribution determine the scaling strategy to use.

Feature transformation: In other circumstances, the initial characteristics might not accurately capture the underlying data patterns or might be challenging for machine learning algorithms to comprehend. To develop new features or alter existing ones, feature transformation techniques including logarithmic transformation, square root transformation, and polynomial expansion can be used. The challenge and the data distribution determine the transformation strategy to choose.

Handling imbalanced data: When the distribution of classes in the target variable is not uniform, there is imbalance in the data. The bulk of network traffic examples are frequently normal in the context of anomaly detection, with a comparatively small amount of malicious traffic. Biased models that benefit the dominant class can be produced from unbalanced data. The class distribution in the dataset can be balanced using methods like random oversampling, random under sampling, or synthetic minority over-sampling technique (SMOTE).

Data partitioning: Typically, the dataset is split into distinct training and testing sets to enable the models to be assessed on unlabelled data. This procedure is crucial for determining how generalizable the created models are and avoiding overfitting. Random splitting, stratified splitting (to preserve the class distribution throughout the splits), and time-based splitting are examples of common data partitioning techniques (particularly relevant for time-series or streaming data).

3.1.3 Feature Selection

As it determines the features that are most pertinent to the precise identification of network anomalies, feature selection is a crucial phase in the creation of an anomaly detection system. Feature selection can enhance model performance by lowering the number of features utilised in the model, reducing overfitting, and lowering computing complexity. The following methods were used in the feature selection procedure in this study:

i. Correlation analysis: In order to determine the linear relationship between characteristics and the target variable, Pearson's correlation coefficient was determined (i.e., normal, or malicious traffic). This study made it possible to find features that are highly connected and can provide the model significant predictive ability. Low correlation features were deleted from the dataset since they were deemed to be less significant.

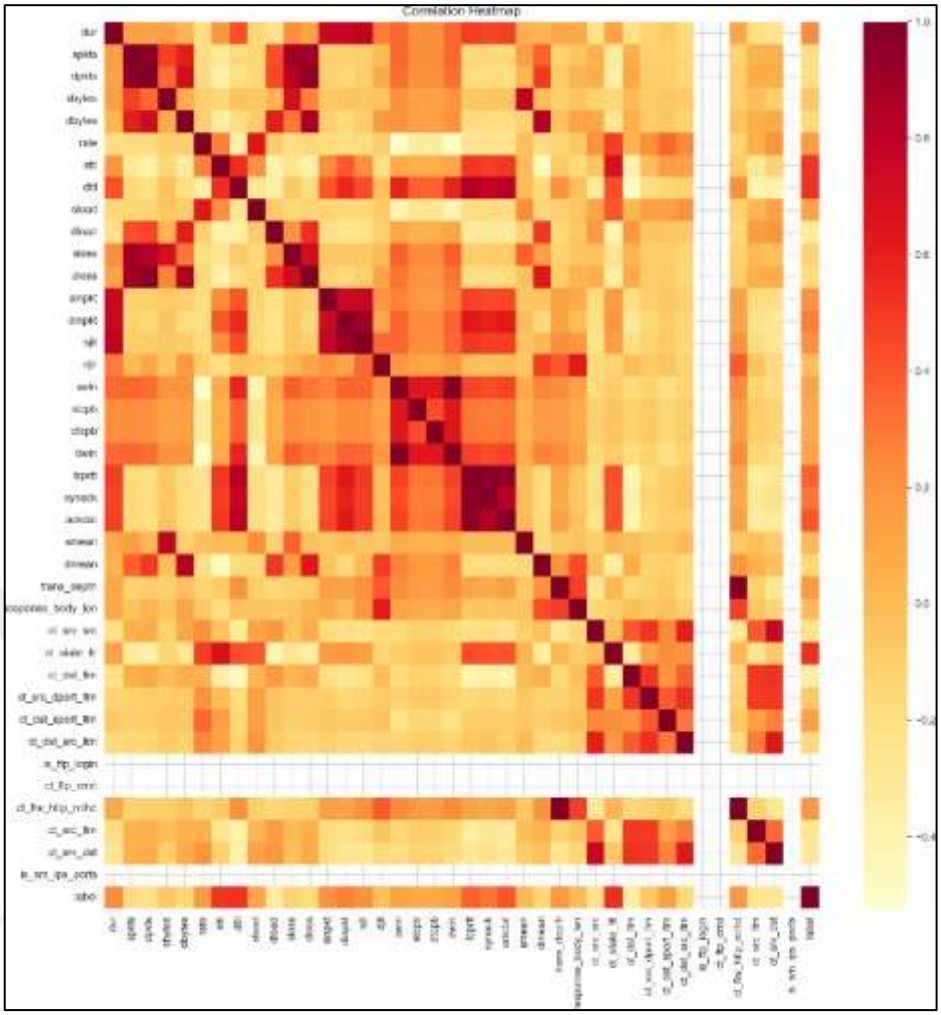

Figure 3.2: Correlation Analysis.

ii. Mutual information: The measurement of the dependence between characteristics and the target variable is done using the mutual information technique. It measures the degree of knowledge acquired about the target variable from being aware of the value of a certain aspect. High mutual information suggests that the feature is crucial for anomaly identification since it shows a strong association between the feature and the target variable. The dataset was cleaned up by removing features with low mutual information.

iii. Recursive feature elimination (RFE): RFE is a backward selection technique that, based on the performance of a selected estimator, iteratively removes the least significant features (e.g., a classification algorithm). The least significant feature is eliminated after each iteration once the estimator has been trained on the entire

set of features initially. Up until the necessary number of features is attained, this process is repeated. The most pertinent features that significantly influence model performance can be found via RFE.

iv. Feature importance from tree-based models: Based on the frequency and efficiency of feature splits in the tree structure, tree-based models, such as decision trees or random forests, can offer an estimate of feature relevance. High-importance features are utilized to split the data more frequently and successfully, which suggests that they have a considerable impact on the model's performance. This technique can be used to rank traits and choose the most crucial ones for additional examination.

v. Wrapper methods: To find the most informative feature set, wrapper techniques entail training a certain model using several subsets of features and assessing the model's performance. Forward selection, backward elimination, and exhaustive search are examples of common wrapper approaches. These techniques can be computationally expensive, especially when working with large feature sets, but because they take the model in use into account, they can produce results for feature selection that are more precise

3.2 MODEL CHOICE

For testing, a range of machine learning algorithms, including both conventional and deep learning methods, were chosen. supervised learning techniques such as Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), Random Forests, and Gradient Boosting Machines were included in traditional algorithms (GBM) and Artificial neural networks (ANN).

3.2.1 Support Vector Machines (SVM)

Support Vector Machine [Alp20] is a highly accurate supervised learning method used for both classification and regression tasks. SVM aims to find a hyperplane in an n-dimensional space where each observation is considered as a point in the space. The hyperplane acts as a decision boundary for separating the observations belonging to different classes.

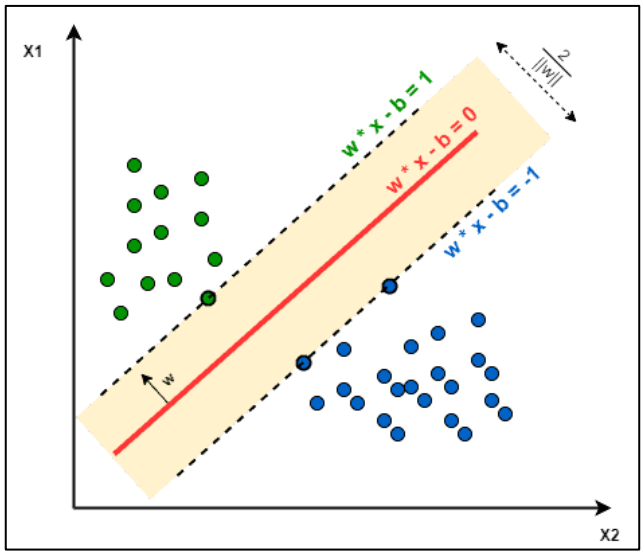

Figure 3.3: Support Vector Machines with Their Support Vectors [55].

Multiple hyperplanes exist for separating the training set of observations but the SVM learning process aims to find the hyperplane that has the maximum margin to improve generalization accuracy. Appropriate Kernel functions are applied to transform a non-linearly separable space into a high dimensional linearly separable space to define maximal margin separating hyperplane. Unknown examples are then mapped onto that space to predict the class based on the side of the decision boundary on which they fall. By selecting the best hyperplane having a larger margin, the possibility of misclassification of unknown examples can be reduced. The scenario of SVM is depicted in Figure 3.3.

3.2.2 K-Nearest Neighbors

KNN [20] is a supervised learning algorithm which is non-parametric as it doesn't make any generalizations and assumptions on the distribution of training data. KNN doesn't build an explicit model, instead it relies on the majority class label of the neighboring examples for classifying a given example. Hence, it is considered as a lazy learner and instance-based classifier. For each new data item X, to be classified into one of the known class labels, the KNN algorithm estimates the nearness of the data item X from all the training sets of observations and considers only its K nearest observations for determining the class of X. The nearness between a pair of observations can be calculated by using the distance metrics. This approach renders KNN particularly effective for classification tasks involving nonlinear

decision boundaries and intricate data distributions, offering simplicity and interpretability in its implementation.

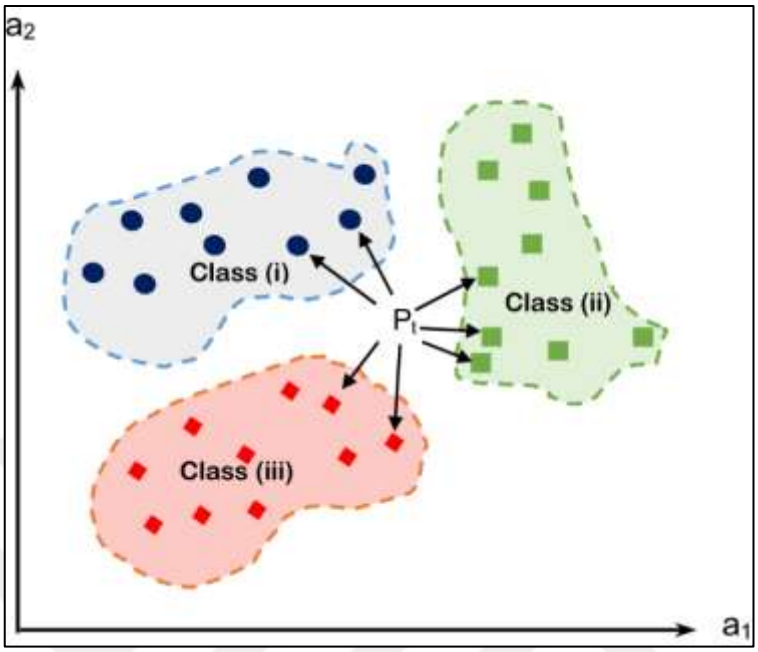

Figure 3.4 : K-Nearest Neighbors [56].

When training the algorithm, the input data and classes are stored. There are mutliple methods to compute the distance between data. Euclidean distance can be used for continuous data. For discrete variables another metric can be used, such as the Hamming distance.

3.2.3 Random Forests

A random forest classifier is a model that is built out of multiple independent decision trees. A decision tree is a very simple tool that can be used to classify a feature vector. An example of a decision tree is shown in Figure 3.5 where the decision tree tries to identify which piece fruit it observes.

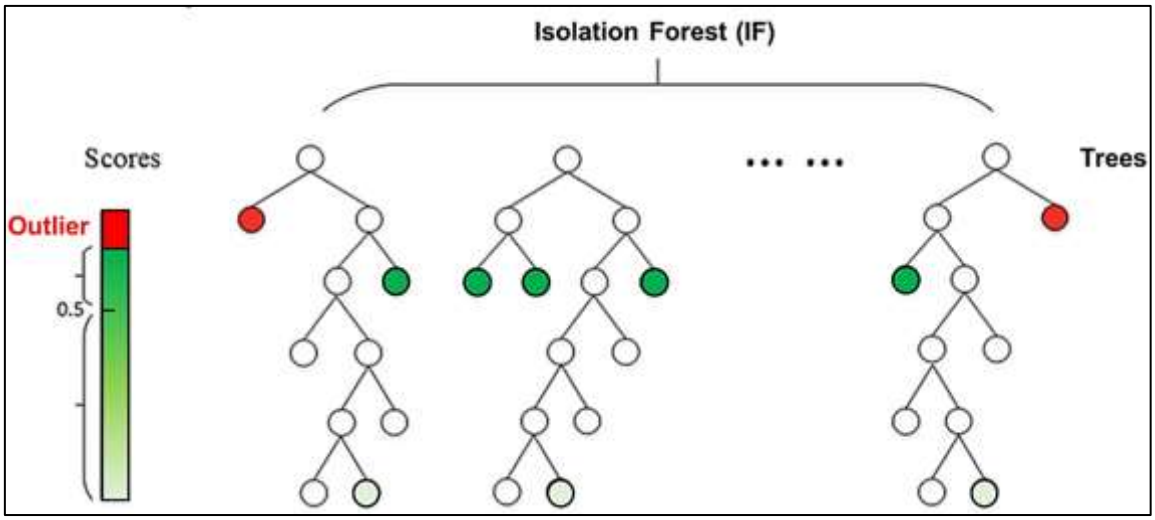

Figure 3.5: Example of a Random Forest Tree.

The idea behind a random forest is to use multiple such decision trees together and the classification label that gets the most votes becomes the decision of the random forest. So, multiple decision trees work together to come to one conclusive decision. It is, however, important that the decision trees are independent (e.g. by using a different set of attributes, different conditions in each decision node, etc.). Random forests generally outperform single decision trees because an error of one or of some of the forest's decision trees does not necessarily make the decision of the forest erroneous. As long as there are more decision trees making a good decision than there are decision trees making a wrong decision, the final decision of the forest will still be correct. This concept is called the wisdom of crowds, in which the decision trees protect other trees from making individual mistakes [57].

3.2.4 Gradient Boosting Machine

Gradient Boosting Machine (GBM) is an ensemble learning technique that sequentially combines weak learners, typically decision trees, to create a powerful predictive model. It minimizes a specified loss function by iteratively fitting new models to the residuals of the combined predictions of previous models, using gradient descent to update parameters. In the context of intrusion detection, GBM analyzes features extracted from network traffic or system logs to distinguish between normal and malicious activity [58]. By learning from labeled data during the training phase, the GBM model becomes adept at identifying anomalous patterns indicative of intrusions. Upon deployment, it assesses incoming data and

raises alerts when detecting deviations from expected behavior, contributing to robust cybersecurity defenses.

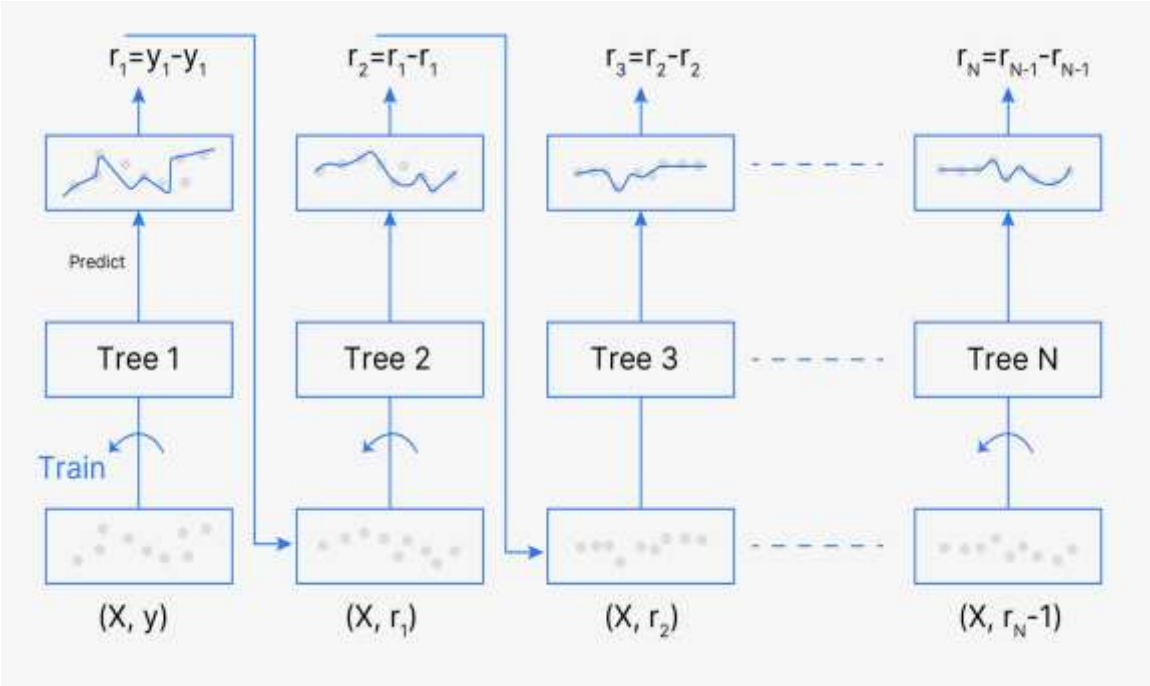

Figure 3.6: Gradient Boosting Machine.

3.2.5 Artificial Neural Networks (ANN)

Neural network-based deep learning approaches have demonstrated effectiveness in a range of applications, including security [80] autonomous vehicle control systems, and biometric identification [79]. Unfortunately, such applications are vulnerable to manipulated data or synthetic inputs. Model accuracy was greatly affected by a change in the input data and these modified samples are called adversarial examples. Intruders can manipulate samples to interfere with both authorized and unauthorized access to applications by exploiting the biometric based authentication framework [61]. Deep neural network models' weakness in image classification tasks was initially proposed by [52]. An adversarial example can be generated when a perturbation is added to the original image, however the deep neural network can confidently produce false findings even if the perturbation is small. The network's recognising capabilities are negatively impacted.

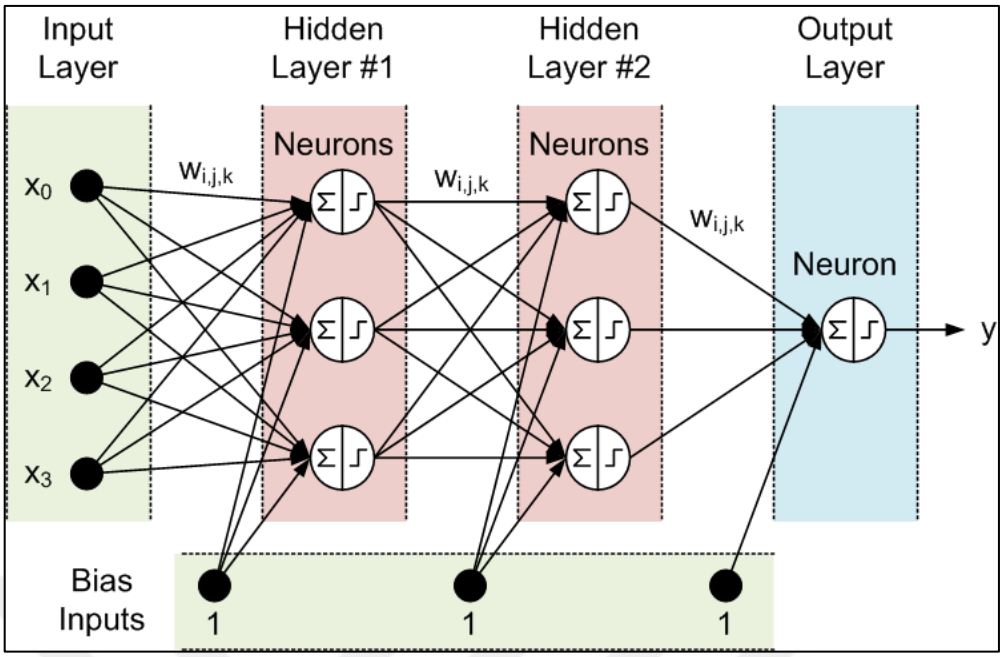

Figure 3.7: A Neural Network Showing the Different Layers.

3.3 PERFORMANCE CRITERIA

Confusion matrix is a table that visually represents the actual and predicted values, it was used to calculate and show the results of the experiments. The output metrics of a confusion matrix provide crucial insights into the performance of a classification model. "True positive" signifies the accurate identification of positive instances, while "True negative" denotes the correct classification of negative instances. Conversely, "False positive" indicates instances that were mistakenly classified as positive when they are actually negative, and "False negative" represents instances incorrectly classified as negative when they are positive. The confusion matrix, where the table showcases the counts of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) instances, aiding in the evaluation and assessment of the model's classification accuracy and performance. After getting the output TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) from the table, the authors calculated the accuracy for each model. Accuracy of an model was calculated according to the following form by dividing the correctly classified images (TP+TN) by the total number of images (TP+TN+FP+FN):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.1)$$

$$Precision = \frac{TP}{TP+FP} \quad (3.2)$$

Recall, also known as sensitivity or true positive rate, is a vital performance metric in classification tasks, particularly in scenarios where correctly identifying positive instances holds significant importance:

In the equations above, True Positive (TP) represents the count of correctly predicted positive cases, where the model correctly identifies positive instances as positive. True Negative (TN) denotes the accurate prediction of negative cases, indicating instances where the model correctly identifies negative samples as negative. Conversely, False Negative (FN) signifies the erroneous prediction of negative cases, where the model incorrectly classifies negative instances as positive, representing a type two error. False Positive (FP) reflects the inaccurate prediction of positive cases, where the model wrongly identifies positive instances as negative, representing a type one error

$$Recall = \frac{TP}{TP+FN} \quad (3.3)$$

$$F1 - Score = \frac{2*Accuracy*Recall}{Accuracy+Recall} \quad (3.4)$$

3.4 CONCLUSION

Though traditional ML classifiers are successful to deal with known attacks, they are severely challenged by the zero-day attacks due to lack of labeled data describing the features specific to the newly devised zero-day attack. However, the collaborative IDSs of an Intrusion Detection Network share information related to zero-day attacks as soon as it was experienced by one or more nodes in the network, thus converting it into a 'new attack' rather than a zero-day attack for the collaborative nodes of the IDN.

4. RESULTS AND DISCUSSION

4.1 PREPARATION OF DATA IN THE EXPERIMENTAL ENVIRONMENT

For this work, around 35 publications are carefully analyzed and an experiment is conducted on a Windows PC from HP with a dual-core processor and 8 GB of RAM. The workbench supplied by an Anaconda platform is called Jupyter, and it is the environment on which the application was developed. Python was utilized as the programming language to create this intrusion detection model. UNSW NB15, a dataset with a 68 percent training set and a 32 percent testing set that consists of 175,341 tuples in the train set and 82,332 tuples in the test set, was used for this study. We preprocessed the data first in order to use it to make wise conclusion Then we applied feature selection to pick high rank features. We have applied supervised machine learning algorithm to classify the data instances into the attacks and normal.

4.2 SOFTWARE ENVIRONMENT

Software Requirements 1. Operating system: Windows XP or higher 2. Technology used : Python 3 3. Tools used: Anaconda, Scikit, Jupyter Notebook Hardware Requirements 1. Processor: intel core i5 2. Motherboard: Genuine Intel 3. RAM: Min 8 GB 4. Graphic card: NVIDIA 940 mx or AMD A8.

4.3 DATA DISTRIBUTION

In Data Preprocessing first the version check is done through which we can get to know which version we are using, after version check we load the dataset i.e., UNSW NB 15 Dataset. Once dataset is loaded, we use LabelEncoder through which dummies are created. Once LabelEncoder is done by One Hot-encoding is used to convert all features to numbers. To prevent features having high values from affecting the results unfairly, the features are scaled. One hot encoding technique is used for transforming categorical information into a numerical format that ML algorithms may use to perform better at prediction

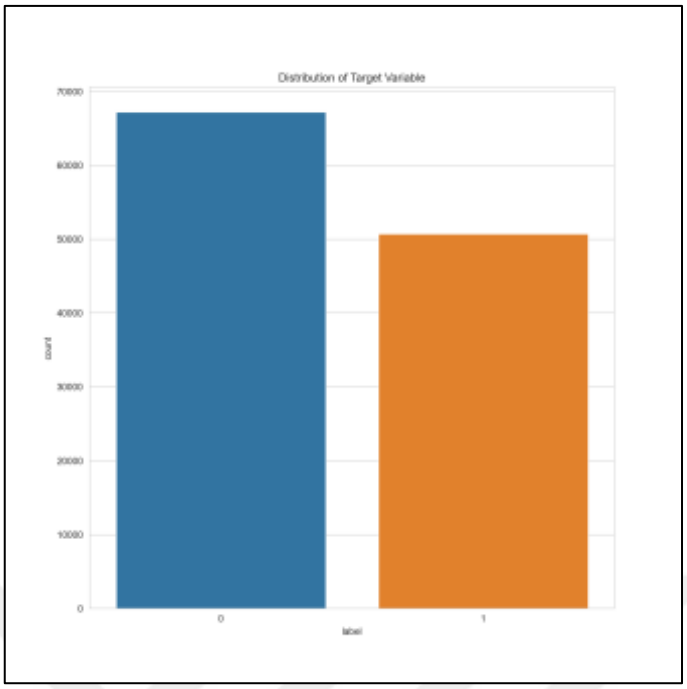

Figure 4.1: UNSW-NB15 Dataset Distribution.

Data pre-processing should be done to prepare the data before building a classifier. Pre-processing is the task of cleaning and transforming the dataset and making it ready for model construction. Models constructed without proper pre-processing may not give reliable results [17]. Pre-processing generally includes the following tasks:

a. Handling Missing Values: Data may include some missing values; those values should be handled either by removing them or replacing them with some suitable value like mean.

b. Removing Insignificant Fields: Data may contain some sparse fields dominated by zero or null values and contribute nothing to define a pattern. Those fields should be identified and removed.

c. Handling Mixed Type of Features: Some machine learning algorithms like KNN deals with numeric data only. But real-world network traffic data may also include some categorical features. So, these categorical features need to be transformed into a numeric form by adopting proper conversion methods.

d. Data Normalization: Normalization is the process of scaling the descriptive features of the dataset into a uniform range. Min-Max and Zscore are the popular methods used for data normalization. Min-Max normalizes the values of features into a linear scale bounded by a given range and it is suitable for uniformly distributed features. Z-score normalizes the data points from Gaussian distribution to Normal distribution so that most of the values of a feature are close to 0.

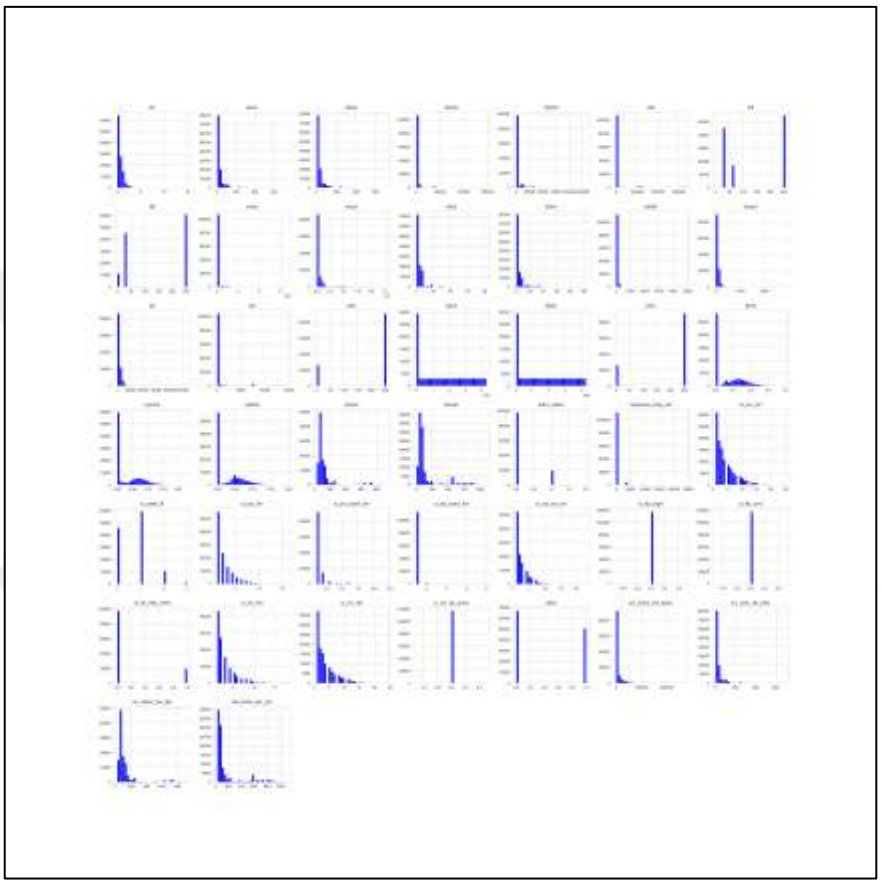

Figure 4.2: Data Distribution by Features.

4.3.1 Normalizing Numerical Features

Numerical feature normalisation is essential for several machine learning techniques. However, it is typically not required for tree-based algorithms like random forests. Multiple decision trees are created using random forests, where decisions are made based on the relative order of feature values rather than their absolute values. As a result, the performance of the model is unaffected by the numerical features' scale.

4.3.2 Data Partitioning

There are training and testing phases in machine learning techniques. To learn the behaviour of traffic over time, mathematical computations are performed on the training dataset in the training step. A test instance is categorized as normal or attacking during the testing process based on the learnt behaviour. Using the UNSWNB15 dataset, we have explored how different ML-based approaches operate. The reduced UNSW NB15 dataset is split into training and testing in proportions of 70% and 30%, respectively. The training and testing datasets respectively for binary classification. The binary class labels are taken from the column containing discrete values.

4.3.3 Model Training and Testing

4.3.3.1 SVM model

The figures 4.3 and 4.4 , and table 4.1 provide the intrusion classification results of the support vector Machine (SVM) in a binary classification. The model shows a efficient performance in terms of AUC value with for the training (0.990703) and test (0.990668) sets are nearly identical, indicating that the model maintains its ability to discriminate between the classes consistently across both datasets. In addition, the model achieved a good F1-Score percentage with for the training (0.955295) and test (0.955204) sets are also very close. More ever the model shows a very False Alarm Rate which used to measure the percentage of normal network activities incorrectly classified with both the training (0.009297) and test (0.009332) sets, further confirming the model's robustness and reliability.

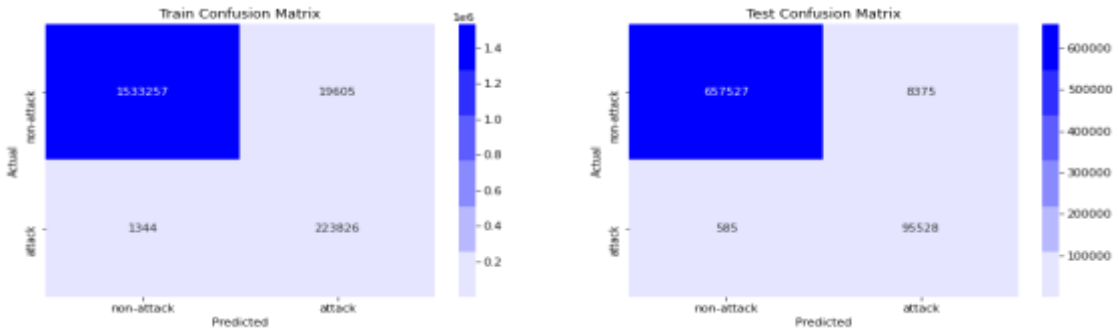

Figure 4.3: SVM Confusion Matrix

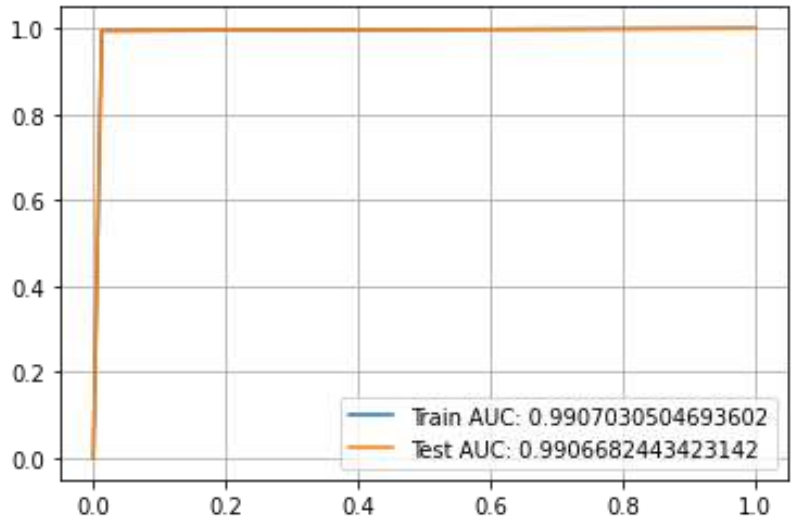

Figure 4.4: Model Performance on Train and Test.

Table 4.1: Classification Report of SVM.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	SVM	99.07%	95.52%	0.0092
Test	SVM	99.06%	95.52%	0.0093

4.3.3.2 KNN model

The KNN model is a supervised machine learning model used for binary classification of intrusion. The performance results of KNN for intrusion detection are illustrated in figure 4.5 and 4.6 and table 4.2. The model exhibit a variable performance, The AUC results in the training stage 0.944084, which show the ability of the model distinguish between normal traffic and attacks. However the model give a law AUC results in the testing set indicating that the model struggles against unseen data. With the same way the F1-Score which represent the balance between precision and recall is good during the training set but it drops down in the testing set. More important criteria is False Alarm Rate which consider high at 0.055916 for the training set and its more higher to 0.081677 for the test set. Those results indicate that the KNN model classified incorrectly a lot of normal instance as attacks.

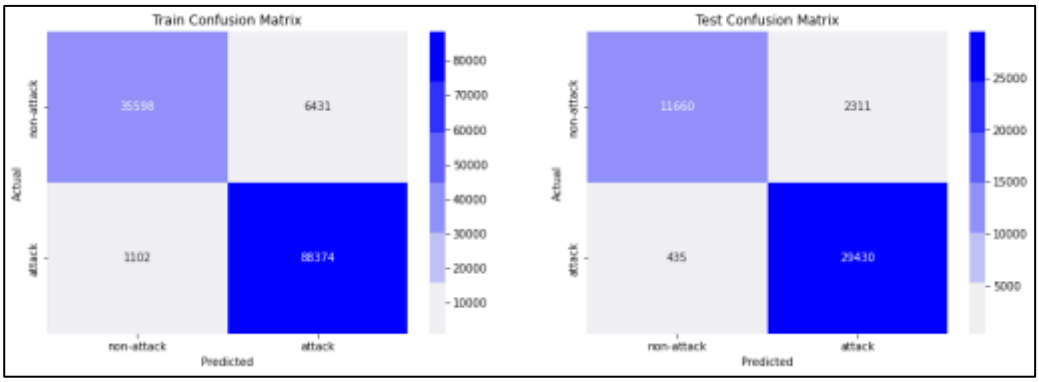

Figure 4.5: KNN Confusion Matrix.

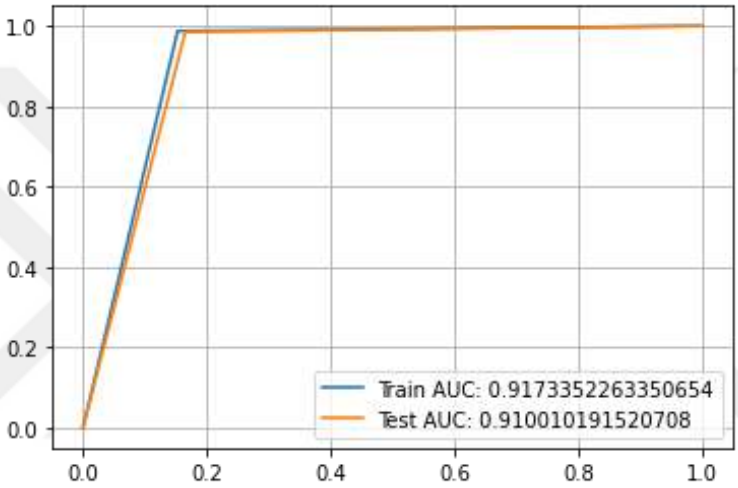

Figure 4.6: Model Performance On Train And Test

Table 4.2: Classification Report of KNN

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	KNN	94.40%	96.84%	0.0559
Test	KNN	91.83%	95.28%	0.0816

4.3.3.3 Random forest

The performance results and classification report of Random Forest model used for intrusion classification and detection in IDS-UNSW-NB15 dataset are illustrated in figure 4.7 and 4.8

and Table 4.3. Initially the model achieved exceptionally high results in the training set with 0.992793, however this value drops during the testing set to 0.985477, those results indicate that the model show excellent performance in detecting unseen data. Additionally , The model achieved a remarkable F1-Score values with 0.989753 on the training set and 0.976750 on the test set. Moreover the model show low False Alarm Rate results with 0.007207 for training and 0.014523 on the test set. Overall the model exhibit an excellent performance during the training and the testing set and keeping the false alarm rate relatively low.

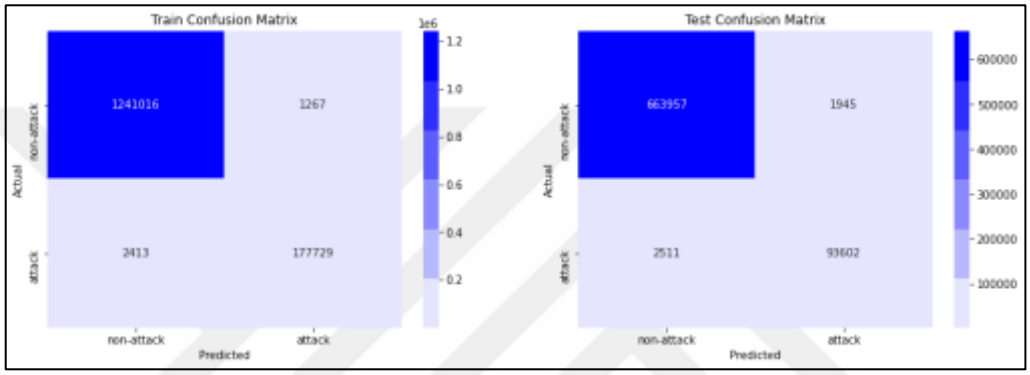

Figure 4.7: RF Confusion Matrix.

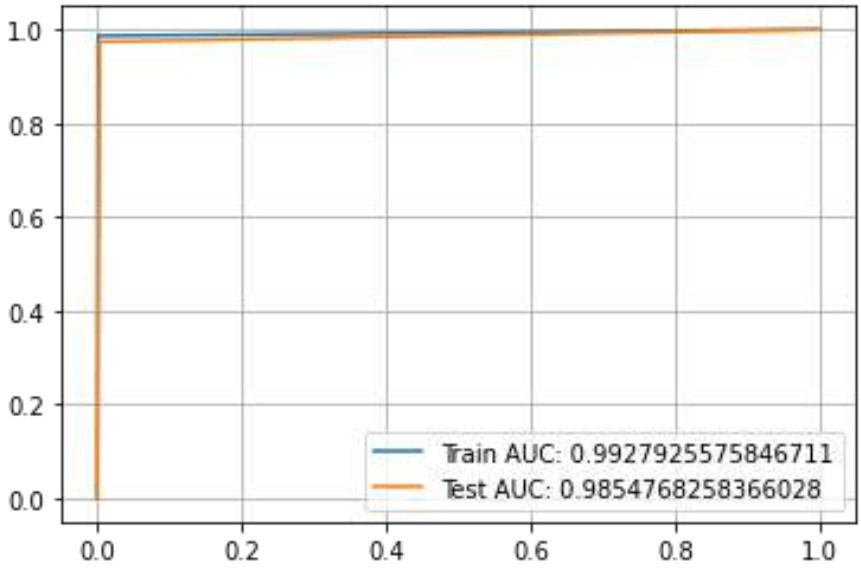

Figure 4.8: Model Performance on Train and Test.

Table 4.3: Classification Report of RF.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	RF	99.27%	98.97%	0.0072
Test	RF	98.55%	97.67%	0.0145

4.3.3.4 Gradient boosting machine

The Gradient Boosting Machine demonstrate a good performance in the training set (See Figure 4.9, Figure 4.10 and Table 4.4) but show some overfitting when applied to the test set. The AUC results exceptionally good with 0.995389 in training set and 0.9864086 in testing set. With the same way the F1-Score exhibit good performance with 0.992080 on the testing set. Moreover the False Alarm rate show a very law results with 0.004611 indicating that the model successfully well classified most of normal traffics.

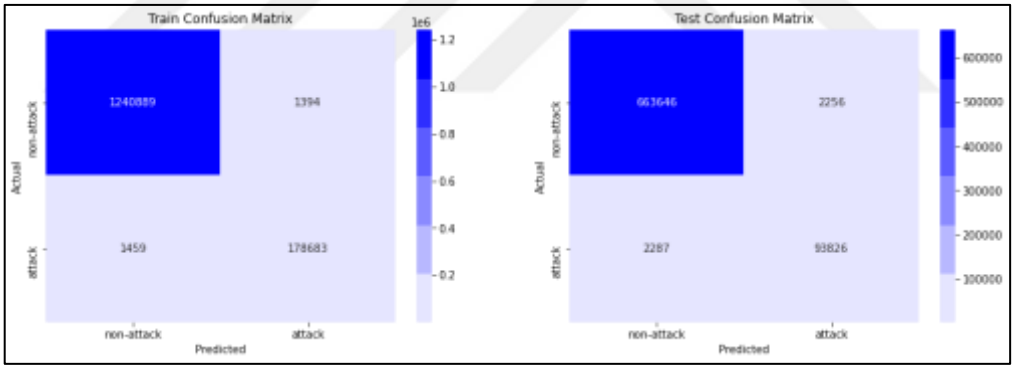

Figure 4.9: XGB Confusion Matrix.

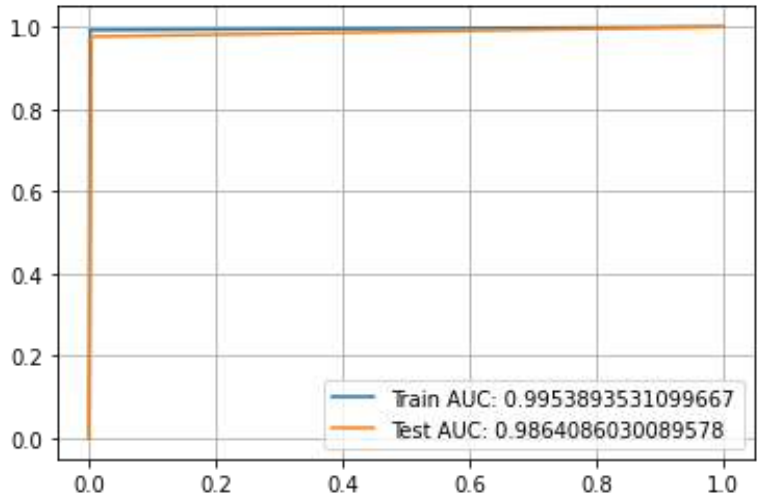

Figure 4.10: Model Performance on Train and Test.

Table 4.4: Classification Report of XGB.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	XGB	99.53%	99.20%	0.0046
Test	XGB	99.64%	97.63%	0.0135

4.3.3.5 Artificial neural networks (ANN)

The promised results in Figure 4.11 and 4.12 and Table 4.5 show the performance results of Artificial Neural Network (ANN) for intrusion classification and detection in UNSW-NB15 dataset, in a binary classification task. As shown , the model exhibit a good generalization capability with AUC value 0.987965 for the training set and 0.987745 for testing set .Additionally , the model achieve approximately a similar results for F1-Score with 0.963481 on the training set and 0.962851 on the test set. Moreover the model achieved a very law False Alarm Rate with training of 0.012035 and a test of 0.012255. Overall, those results indicate that ANN model successfully maintains the same performance in unseen data and avoid very well the overfitting issue.

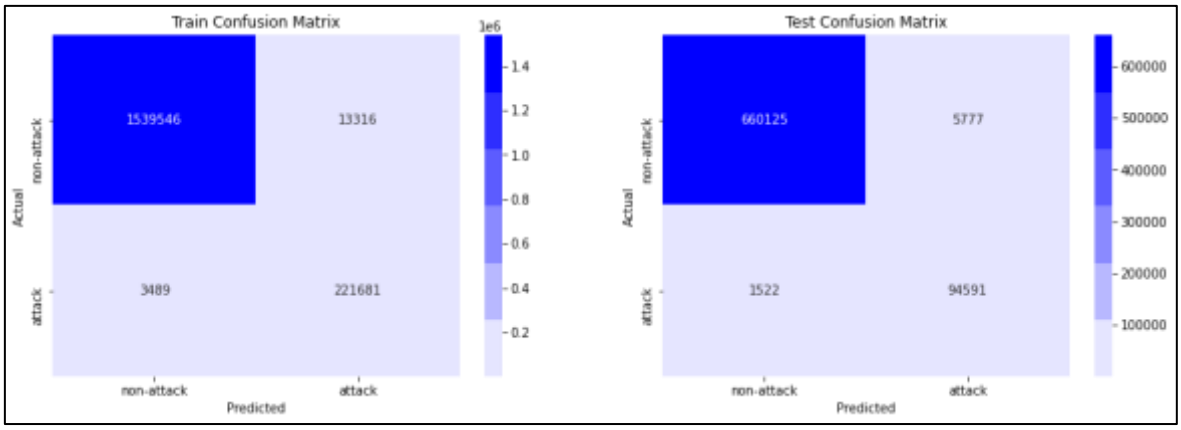

Figure 4.11: ANN Confusion Matrix.

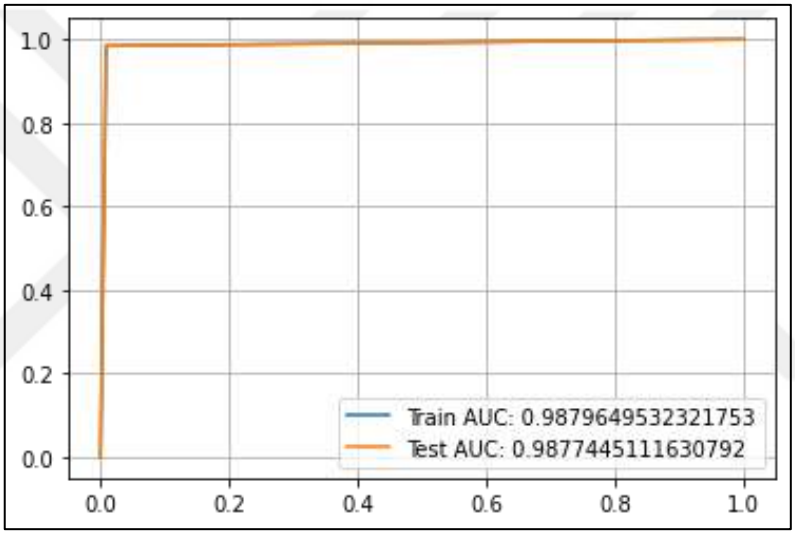

Figure 4.12: Model Performance on Train and Test.

Table 4.5: Classification Report of ANN.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	ANN	98.79%	96.34%	0.0120
Test	ANN	98.77%	96.28%	0.0122

4.3.3.6 Voting classifier model

The classification results of Voting classifier illustrated in Figure 4.13 and 4.14 and Table 4.6 show a remarkable performance. Initially the model achieved an excellent AUC performance with 0.994837 on the training set and 0.986986, demonstrating capability of generalization to unseen data. Also, the model achieved a very law false alarm rate with 0.005163 on training and 0.013014 on testing set. The results recommend that the voting model which leverage multiple models (Previous mentioned models) is enough effective and maintain and efficient performance on the test stage.

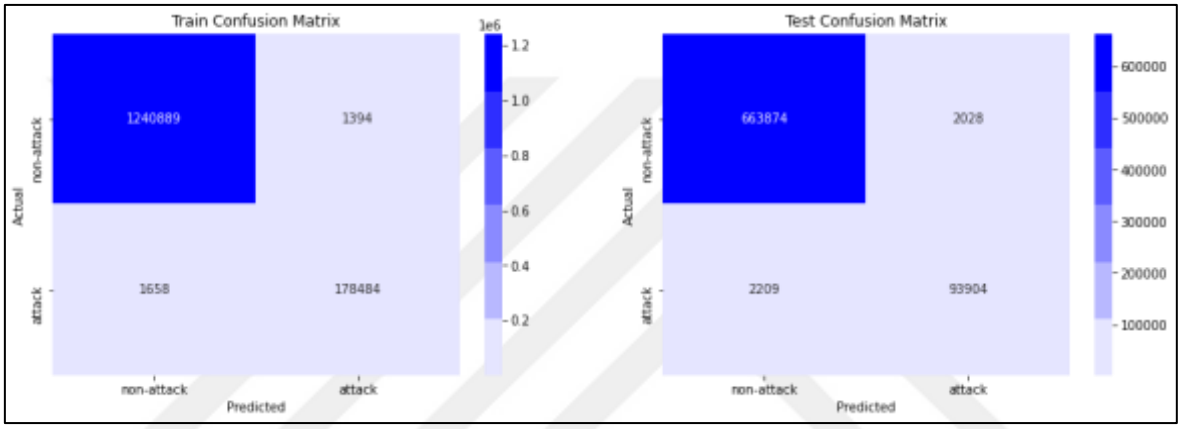

Figure 4.13: Voting Confusion Matrix.

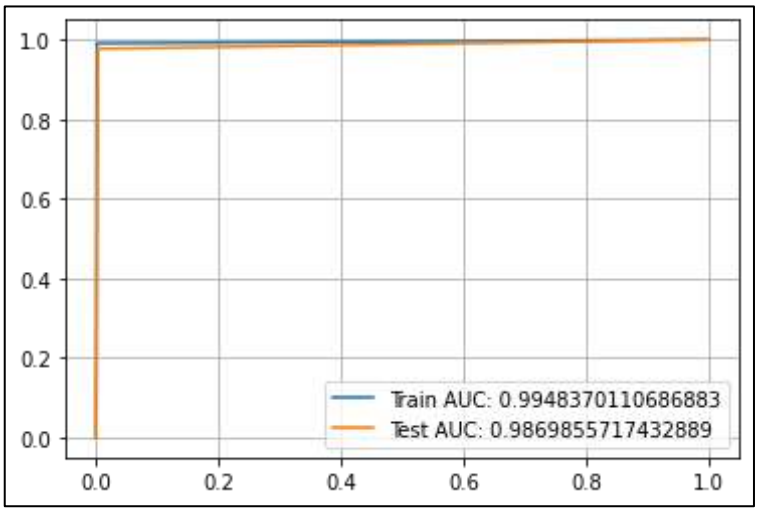

Figure 4.14: Model Performance on Train and Test.

Table 4.6: Classification report of Voting.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	Voting	99.48%	99.15%	0.0051
Test	Voting	98.69%	97.79%	0.0130

4.3.4 Model Optimize

In this intrusion detection solution, we used GridSearch CV for hyperparameters tuning (HOP) of the machine learning models. The results of HOP are variated , where some models maintain similar performance after applying HOP for example KNN model before achieved an AUC value 91.83% and with HOP 91.00%.

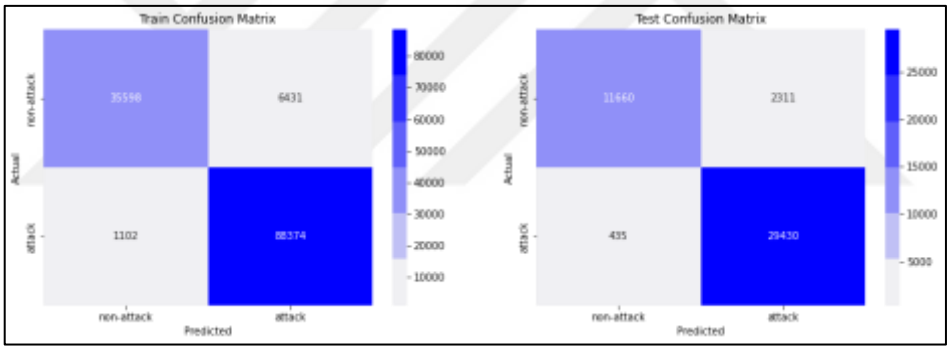

Figure 4.15: KNN Confusion Matrix with CV**.**

Table 4.7: Classification Report of KNN with CV.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	KNN	91.73%	95.91%	0.0826
Test	KNN	91.00%	95.54%	0.0899

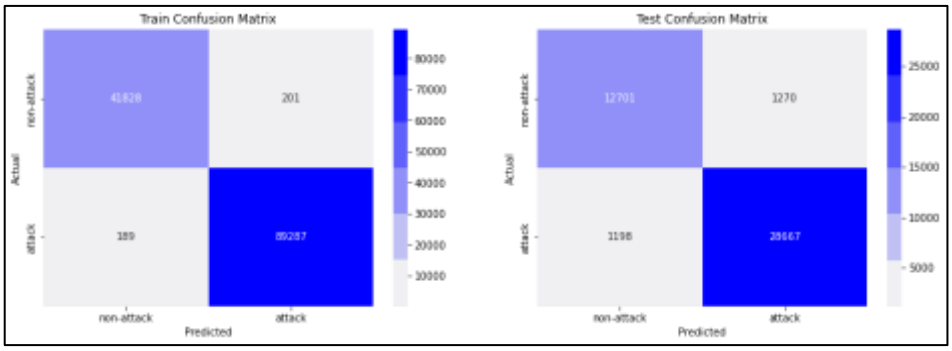

Figure 4.16: ANN Confusion Matrix with CV.

Table 4.8: Classification Report of ANN with CV.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	ANN	99.65%	99.78%	0.0034
Test	ANN	93.44%	95.87%	0.0655

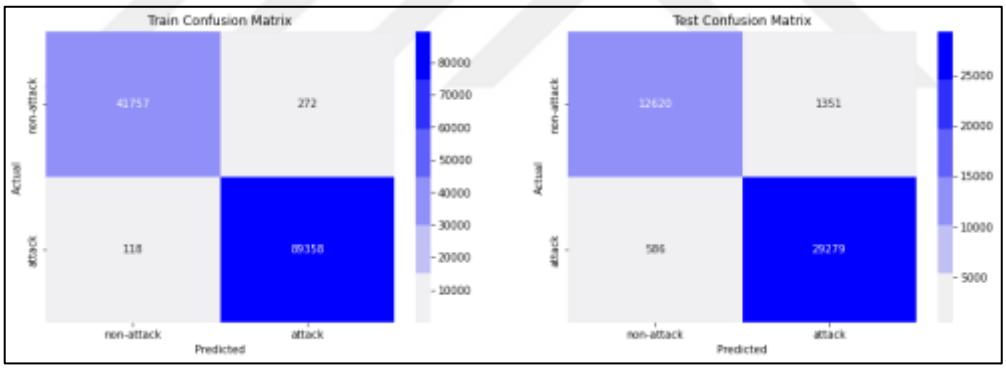

Figure 4.17: RF Confusion Matrix with CV.

Table 4.9: Classification Report of RF with CV.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	RF	99.61%	99.78%	0.0038
Test	RF	94.18%	96.79%	0.0581

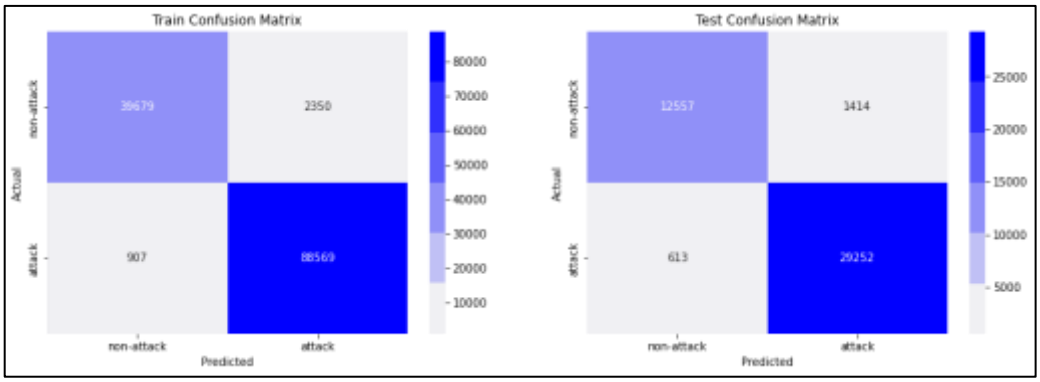

Figure 4.18: XGB Confusion Matrix with CV.

Table 4.10: Classification Report Of RF With CV.

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	XGB	96.69%	98.19%	0.0330
Test	XGB	93.91%	96.65%	0.0608

4.4 MODEL COMPARISON

The following results in Figure 4.20 display the comparison results between the machine learning model used for the classification and detection of intrusions in UNSW-NB15 dataset for binary classification. A shown the SVM model achieved the AUC values with 99.07%. Similarly voting and ANN models show a good AUC value with 98.79% and 98.77% respectively. Whereas KNN model give the lowest AUC value with 94.40%.

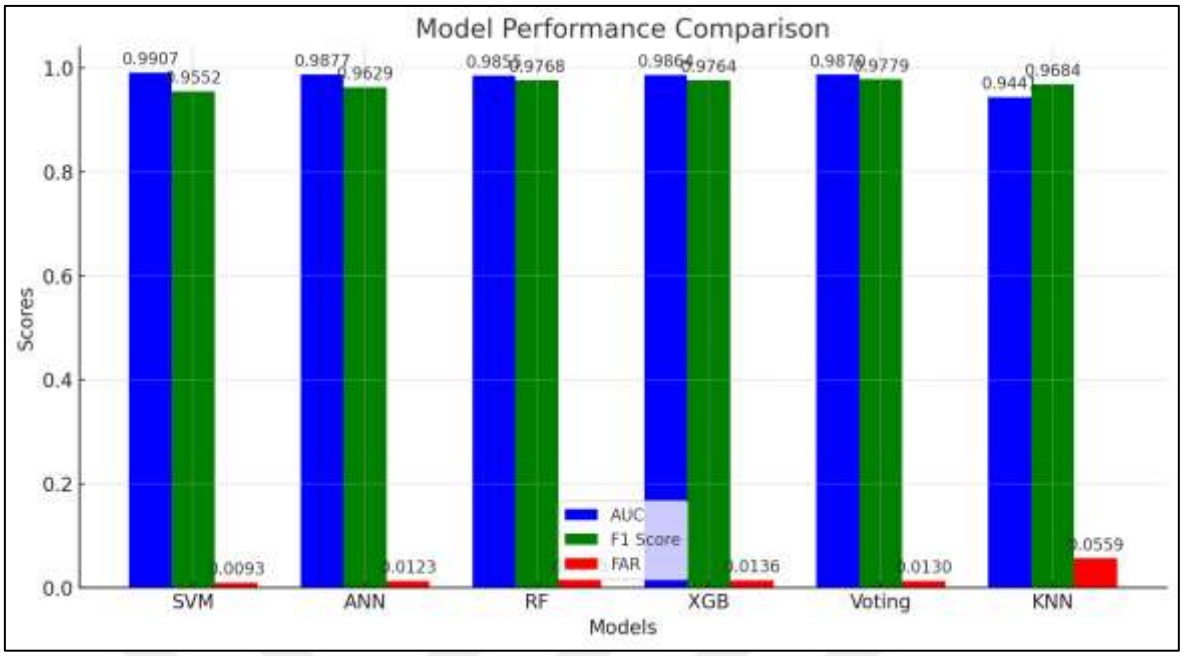

Figure 4.19: Models Performance Comparison

4.5 CONCLUSION

The chapter proposed an automatic detection and classification of Intrusion recognition system. The source dataset is UNSW-NB15 datasets. The performance analysis is carried out with respect to IDS in terms of accuracy. This chapter proposed statistical methodology in identifying Intrusion using Labeled and Unlabeled datasets. The proposed methodology stated in this chapter achieves 99.07% of AUC with SVM in labeled classification and 97.79% F1-Score with ensemble voting model.

5. CONCLUSIONS AND FUTURE WORKS

5.1 CONCLUSION

An increase in internet dependency in all walks of life, the digital nature of data in huge amounts getting accumulated through online transactions and decentralization of data repositories, have led to the growing demand for the development of effective security algorithms. Intrusion Detection Systems (IDSs) play a key role in protecting cyberspace by going through the entire details of the traffic packet to detect intrusions. Machine Learning (ML) approaches are widely used for IDS development as it needs to analyze extensive amounts of data. Machine learning offers many algorithms that can analyze huge amounts of data and helps IDS in taking better decisions.

5.2 FUTURE WORKS

Some future extensions are identified to carry out the further research and are mentioned below.

a. This research develops separate intrusion detection systems to detect known, new and zero-day attacks. A common framework can be developed as a future extension for detecting known, new and zero-day attacks.

b. The zero-day attack detection framework can be extended further for various attack domains, such as malware detection.

c. Deep Feed Forward Neural Network is used in the zero-day attack detection framework. The work can be extended by checking the possibility of applying other deep learning algorithms to further enhance the zero-day attack detection accuracy.

d. Since, few target labels are generated with the ‘target soft label generation’ phase of the zero-day attack detection framework, applicability of Weighted Distance KNN classifier that gives more weightage for target instances can be investigated for detecting zero-day attacks.

REFERENCES

[1] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, “Flow-based benchmark data sets for intrusion detection,” in Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS). ACPI, 2017, pp. 361–369.

[2] A. Verma and V. Ranga, “On evaluation of network intrusion detection systems: Statistical analysis of CIDDS-001 dataset using machine learning techniques,” Pertanika Journal of Science and Technology, vol. 26, no. 3, pp. 1307–1332, 2018.

[3] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, “DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber-Physical Systems,” IEEE Transactions on Industrial Informatics, vol. 17, no. 8, pp. 5615–5624, 2021.

[4] StatisticsTimes, “Projected GDP Ranking,” 2021. [Online]. Available: https: //statisticstimes.com/economy/projected-world-gdp-ranking.php

[5] C. MacKechnie, “Information Technology & Its Role in the Modern Organization,” 2019. [Online]. Available: https://smallbusiness.chron.com/information-technologyits-role-modern-organization-1800.html

[6] B. Whitaker, “SolarWinds: How Russian spies hacked the Justice, State, Treasury, Energy and Commerce Departments,” 2021. [Online]. Available: https://www. cbsnews.com/news/solarwinds-hack-russia-cyberattack-60-minutes-2021-02-14/

[7] L. Constantin, “SolarWinds attack explained: And why it was so hard to detect,” 2020. [Online]. Available: https://www.csoonline.com/article/3601508/solarwindssupply-chain-attack-explained-why-organizations-were-not-prepared.html

[8] M. Brennan, “”Hack everybody you can”: What to know about the massive Microsoft Exchange breach,” 2021. [Online]. Available: https://www.cbsnews.com/ news/microsoft-exchange-server-hack-what-to-know/

[9] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, “Internet of Things intrusion Detection: Centralized, On-Device, or Federated Learning?” IEEE Network, vol. 34, no. 6, pp. 310–317, 2020.

[10] A. Muallem, S. Shetty, L. Hong, and J. W. Pan, “TDDEHT: Threat Detection Using Distributed Ensembles of Hoeffding Trees on Streaming Cyber Datasets,” Proceedings - IEEE Military Communications Conference MILCOM, vol. 2019-Octob, pp. 219–224, 2019.

[11] T. H. Hai and N. T. Khiem, “Architecture for IDS Log Processing using Spark Streaming,” in Proc. of the 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE), no. June, 2020, pp. 3–7.

[12] K. Alrawashdeh and C. Purdy, “Fast hardware assisted online learning using unsupervised deep learning structure for anomaly detection,” 2018 International Conference on Information and Computer Technologies, ICICT 2018, pp. 128–134, 2018.

[13] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.

[14] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings et al., “Advances and open problems in federated learning,” Foundations and Trends® in Machine Learning, vol. 14, no. 1–2, pp. 1–210, 2021.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50–60, 2020.

[16] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Koneˇcn´y, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, “Towards federated learning at scale: System design,” 2019.

[17] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," IEEE Journal on Selected Areas in Communications, vol. 37, no. 6, pp. 1205–1221, 2019.

[18] J. Koneˇcn´y, H. B. McMahan, F. X. Yu, P. Richt´arik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2017.

[19] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023.

[20] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020.

[21] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," 2020.

[22] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," Advances in Neural Information Processing Systems, vol. 33, pp. 3557–3568, 2020.

[23] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in ICISSP 2018, 01 2018, pp. 108–116.

[24] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, "Kdd cup 99 data sets: A perspective on the role of data sets in network intrusion detection research," Computer, vol. 52, no. 2, pp. 41–51, 2019.

[25] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. Habibi Lashkari, "Detection of doh tunnels using time-series classification of encrypted traffic," in 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2020, pp. 63–70.

[26] R. K. Vigneswaran, R. Vinayakumar, K. Soman, and P. Poornachandran, “Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security,” in 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018, pp. 1–6.

[27] A. L. Maas, A. Y. Hannun, A. Y. Ng et al., “Rectifier nonlinearities improve neural network acoustic models,” in Proc. icml, vol. 30. Citeseer, 2013, p. 3.

[28] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, “Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection,” IEEE Access, vol. 6, pp. 1792–1806, 2018.

[29] S. Zaman and F. Karray, “Lightweight ids based on features selection and ids classification scheme,” in 2009 International Conference on Computational Science and Engineering, vol. 3, 2009, pp. 365–370.

[30] Y. Gong, S. Mabu, C. Chen, Y. Wang, and K. Hirasawa, “Intrusion detection system combining misuse detection and anomaly detection using genetic network programming,” in 2009 ICCAS-SICE, 2009, pp. 3463–3467.

[31] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, “Deep learning approach for intelligent intrusion detection system,” IEEE Access, vol. 7, pp. 41 525–41 550, 2019.

[32] I. Yeo and R. A. Johnson, “A new family of power transformations to improve normality or symmetry,” Biometrika, vol. 87, no. 4, pp. 954–959, 12 2000. [Online]. Available: https://doi.org/10.1093/biomet/87.4.954

[33] V. A. Mankov, V. Y. Deart, and I. A. Krasnova, “Evaluation of the effect of preprocessing data on network traffic classifier based on ml methods for qos predication in real-time,” in Advances in Artificial Systems for Medicine and Education IV, Z. Hu, S. Petoukhov, and M. He, Eds. Cham: Springer International Publishing, 2021, pp. 55–64.

[34] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, “Training deep neural networks on imbalanced data sets,” in 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 4368–4374.

[35] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, “Cost-sensitive learning of deep feature representations from imbalanced data,” IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 8, pp. 3573–3587, 2018.

[36] A. Yulianto, P. Sukarno, and N. A. Suwastika, “Improving adaboost-based intrusion detection system (ids) performance on cic ids 2017 dataset,” in Journal of Physics: Conference Series, vol. 1192, no. 1. IOP Publishing, 2019, p. 012018.

[37] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, “Ddosnet: A deep-learning model for detecting network attacks”, in 2020 IEEE 21st International Symposium on” A World of Wireless, Mobile and Multimedia Networks”(WoWMoM), pp. 391–396, 2020.

[38] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection”, IEEE transactions on emerging topics in computational intelligence, vol 2, no 1, pp. 41–50, 2018.

[39] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. AlNemrat, and S. Venkatraman, “Deep learning approach for intelligent intrusion detection system”, IEEE Access, vol 7, pp 41525–41550, 2019.

[40] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, “An intrusion detection model based on feature reduction and convolutional neural networks”, IEEE Access, vol 7, pp 42210–42219, 2019.

[41] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, “An empirical comparison of repetitive undersampling techniques”, in 2009 IEEE international conference on information reuse integration, pp 29–34, 2009.

[42] Kamalakanta Sethi, Rahul Kumar, Nishant Prajapati, and Padmalochan Bera. Deep reinforcement learning based intrusion detection system for cloud infrastructure. In

2020 International Conference on COMmunication Systems NETworkS (COMSNETS), pages 1–6, 2020.

[43] Saeid Soheily-Khah, Pierre-François Marteau, and Nicolas Béchet. Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process: A case study on the iscx dataset. In 2018 1st International Conference on Data Intelligence and Security (ICDIS), pages 219–226, 2018.

[44] S Tümer and E Türkmen. A study on signature-based intrusion detection systems. International Journal of Advanced Computer Science and Applications, 2011.

[45] D'hooge L. Wauters T. et al. Verkerken, M. Towards model generalization for intrusion detection: Unsupervised machine learning techniques. volume J Netw Syst Manage 30, 2022.

[46] Shashank Vohra and Harshal Shah. Kubernetes: A comprehensive overview. IEEE Potentials, 2018.

[47] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection", IEEE Access, vol 6, pp 52843–52856, 2018.

[48] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study", Journal of Information Security and Applications, vol 50, pp 102419, 2020.

[49] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison", Journal of Network and Computer Applications, vol 169, pp 102767, 2020.

[50] L. Deng and D. Yu, "Deep learning: methods and applications", Foundations and trends in signal processing, vol 7, no 3–4, pp 197–387, 2014.

[51] J. L. Leevy and T. M. Khoshgoftaar, “A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data”, Journal of Big Data, vol 7, no 1, pp 1–19, 2020.

[52] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization”, in ICISSp, pp 108–116, 2018.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting”, The journal of machine learning research, vol 15, no 1, pp 1929–1958, 2014.

[54] Y.-F. Hsu and M. Matsuoka, “A Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System”, in 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), pp 1–6, 2020.

[55] J. Li, W. Wu, and D. Xue, “An intrusion detection method based on active transfer learning”, Intelligent Data Analysis, vol 24, no 2, pp 363–383, 2020.

[56] David Boucher and Nick Kratzke. Understanding kubernetes: A comparison of deployment strategies. In Proceedings of the International Conference on Internet Computing, 2019.

[57] Lin Chen, Xiaoyun Kuang, Aidong Xu, Siliang Suo, and Yiwei Yang. A novel network intrusion detection system based on cnn. In 2020 Eighth International Conference on Advanced Cloud and Big Data (CBD), pages 243–247, 2020.

[58] Zouhair Chiba, Noreddine Abghour, Khalid Moussaid, Amina El omri, and Mohamed Rida. Intelligent approach to build a deep neural network based ids for cloud environment using combination of machine learning algorithms. Computers Security, 86:291–317, 2019.

[59] Fabrício S. da Silva, Rafael A. Oliveira, and Paulo F. Pires. Microservices: A systematic mapping study. Journal of Systems and Software, 2019.

[60] Elsayed et al. Network anomaly detection using lstm based autoencoder. volume Q2SWinet ’20, 2020.

[61] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Jose Rubio. An updated performance comparison of virtual machines and linux containers. ACM SIGPLAN Notices, 2014.

[62] Yucheng Han and Ting Chen. Intrusion detection systems: A machine learning approach. International Journal of Machine Learning and Cybernetics, 2015.

[63] Ilhan Firat Kilincer, Fatih Ertam, and Abdulkadir Sengur. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. Computer Networks, 188:107840, 2021.