East West University

"Network Intrusion Detection Using Machine Learning & Deep Learning"

PREPARED BY

Name : Meherunesa Labonno

ID: 2018-2-50-025

Name : Sabbir Ahmed

ID: 2018-3-50-019

SUPERVISED BY

Dr. Mohammad Arifuzzaman

Chairperson, Associate Professor, Department of ECE

This Thesis Paper is Submitted in Partial Fulfillment of the Requirements of the Degree of Bachelor of Science in "Information & Communications Engineering"

DEPARTMENT OF ELECTRONICS & COMMUNICATIONS ENGINEERING
EAST WEST UNIVERSITY

APPROVAL

The thesis titled "Network Intrusion Detection System Using Machine Learning & Deep Learning Algorithms" submitted by Meherunesa Labonno (ID: 2018-2-50-025) and Sabbir Ahmed (ID: 2018-3-50-019) to the Department of Electronics and Communications Engineering, East West University, Dhaka, Bangladesh has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Information and Communications Engineering and approved as to its style and contents.

Approved By

_____

(Supervisor)

Dr. Mohammad Arifuzzaman

Chairperson, Associate Professor

ECE Department

East West University

Dhaka, Bangladesh

DECLARATION

We certify that our work has not been submitted previously for the award of a degree neither this University nor any other University. To the best of my knowledge and belief this thesis contains no information that has already been published or written by someone else except where due reference is made in the thesis itself. We hereby, state that the research we conducted in this thesis paper is the outcome of the investigation performed by us under the supervision of Dr. Mohammad Arifuzzaman, Chairperson, Associate Professor, Department of Electronics & Communications Engineering, East West University, Dhaka, Bangladesh.

Countersigned

(Supervisor)

Dr. Mohammad Arifuzzaman

Signature

Meherunesa Labonno

ID: 2018-2-50-025

Signature

Sabbir Ahmed

ID: 2018-3-50-019

ACKNOWLEDGEMENT

We sincerely thank Dr. Mohammad Arifuzzaman, our supervisor, for his important direction and supervision from the suggested idea through its implementation in our research. His propensity to provide his expertise and time is incredibly reviving. We could also like to thank him for his supportive & interactive behavior and extraordinary inclination. Our supervisor provided us with a wealth of valuable insights and ideas regarding AI throughout the course of our research. His persistent solace gave us the assurance we needed to complete our work. Finally, we express our gratitude to the Almighty and to our supervisor for their uncompromising assistance. Without the wise guidance of our supervisor throughout the research process, this thesis would not have been possible.

ABSTRACT

In recent decades, rapid development in the world of technology and networks has been achieved, also there is a spread of Internet services in all fields over the world. Piracy numbers have increased, also a lot of modern systems were penetrated, so the developing information security technologies to detect the new attack become an important requirement. One of the most important information security technologies is an Intrusion Detection System (IDS) that uses machine learning and deep learning techniques to detect anomalies in the network. The main idea of this paper is to use an advanced intrusion detection system with high network performance to detect the unknown attack package. We use different kind of machine learning algorithm with high accuracy to detect which attack is the most in these dataset. In this paper, DNNs have been utilized to predict the attacks on Network Intrusion Detection System (N-IDS). A DNN with 0.1 rate of learning is applied and is run for 100 number of epochs and KDDCup-'99' dataset has been used for training and benchmarking the network. We compare between both of them on the same dataset .

Table Of Content

CHAPTER FOUR (Machine Learning Algorithm)

CHAPTER FIVE (Deep Learning Algorithm)

CHAPTER SIX (Dataset)

CAPTER SEVEN(Result and Analysis)

CHAPTER EIGHT (Conclusion)

List of Tables

Table No. Title Page

List of Figures

Figure No. Title Page

CHAPTER ONE

INTRODUCTION

Internet usage is very popular, and as a result, threats to the network are also developing quickly. 2.8 billion malware attacks were launched globally in the first half of 2022, according to a Statista survey. The number of malware attacks detected in 2021 was 5.4 billion. The most malware attacks have been found in recent years in 2018, when 10.5 billion of them were reported globally.[1][2]

There are many other types of attacks that can be used, including Brute Force Attacks, Heartbleed Attacks, DoS Attacks, DDoS Attacks, Web Attacks, etc. The network's bandwidth is expanding quickly in response to the increase of internet users. Currently, the normal speed ranges from 1 Gbps to 10 Gbps for a typical data center. For large corporations or big tech firms like Google, Facebook, etc., the download and upload speeds are varied and range from 40 Gbps to 100 Gbps.[4] [3]

A security technology called a network-based intrusion detection system (NIDS) guards against internal and external attacks as well as unwanted network access. which hardware and/or software designed. The most well-known idea is that of a firewall, which is designed to guard against unwanted access by IP address and port number throughout the entire network and to manage these actions using NIDS. Counting the number of network intrusion attempts, such as denial-of-service attacks and other hacking activities that could damage the security of a single computer or the entire network, is one of its many and varied working uses.[5]

NIDS is typically installed outside of the firewall, allowing for the monitoring of all external traffic while simultaneously sensing and identifying any unusual activity.

Types Of An Intrusion Detection System

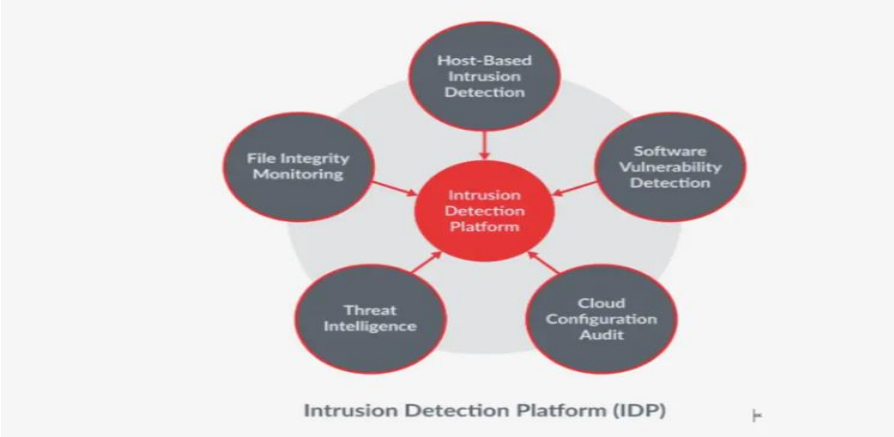

Figure 1: Types of Intrusion detection system

Due to the complexity of the network, it is recommended to choose an NIDS to monitor the changing network environment when a device is linked to a complex network, like one with 1000 nodes. This leads to the conclusion that only one IDS in a network can harm sensitive or confidential data. When we employ DPI (Deep packet Inspection), which matches the pattern against signature packet rules, it would be impossible to process the enormous amount of data due to the network's throughput having only one entry point. This level requires greater processing resources, which could be too much for the NIDS at the moment.

A network bottleneck can easily develop in an NIDS that is overloaded. In this scenario, incoming and outgoing packets may encounter significant delays as a result of the examination of previous packets, or in the worst case scenario, NIDS may even drop the packet. This advantage is simple for an aggressor to use.

Network intrusion detection systems (NIDS) are categorized into three classes: signature-based NIDS (SNIDS), anomaly-based NIDS (ANIDS), and hybrid network intrusion detection. An attacker-specific set of rules is pre-installed in a signature-based NIDS. While the attack is being detected, a pattern matching is done. The traffic shape is used by the anomaly-based NIDS to identify the intrusion. Its goal is to calculate the departure from the flow of traffic. A NIDS with a signature-based approach is well known for its ability to identify any known attack with high accuracy and a low number of false alarms. However, anytime an unidentified attack occurs or there is abnormal network traffic, the performance of signature-based NIDS starts to degrade and becomes difficult. The IDS system comes with pre-installed signatures that must be matched in order to find any attacks.

Although Anomaly Detection NIDS (ADNIDS) generates a disproportionate amount of false positive alarms, it is theoretically possible to identify every unknown attack where this concept is widely accepted by the academic community. Both anomaly-based and signature-based detection are employed by a hybrid IDS. The benefits of both worlds are combined in a hybrid system. A Hybrid Intrusion Detection system can detect both new and old intrusion tactics by looking for trends and isolated incidents. The even greater increase in reported

concerns is the hybrid system's only drawback. Although it's difficult to interpret this increase in alerts as a bad thing given that an IDS's main function is to alert users to potential intrusions, As a result, it is able to identify more potential attacks with a lower mistake rate than when employing each system separately.

We have observed two difficulties while deploying a flexible and effective NIDS for unanticipated future threats. First of all, choosing the right attributes for a network traffic dataset can be challenging for unidentified attacks. The characteristics chosen to identify one type of attack might not be compatible with those chosen to identify another type of attack. When this occurs, NIDS treats the traffic as usual or displays an error (False alarm). The lack of a tagged real-time traffic dataset makes developing an NIDS system difficult. To identify real-time traffic and raw data in relation to various attacks would need an enormous amount of work.

Additionally, a network administrator needs to be secured in order to ensure that all users' privacy is protected as well as the confidentiality of any network, such as an organization's network structure, from attack. The Anomaly Detection NIDS has been detected using a wide range of machine learning algorithms. For illustration

Artificial neural networks (ANN), support vector machines (SVM), random forests, self-organized, naive Bayesian, and deep learning are examples. In order to distinguish any anomaly from regular traffic, Network Intrusion Detection Systems have developed as classifiers. Some NIDS systems have the feature selection functionality to create a subset of relevant attributes from the dataset to enhance classification results.

Since the invention of computer architectures, there has been research on ID in network security. Holistic IDS is increasingly being addressed with ML approaches and solutions, although the training data that is currently available is small and is frequently used mainly for benchmarking. One of the largest publicly accessible databases is the one from DARPA. The 1999 KDD-Cup competition was held during the Fifth International Conference on Knowledge Discovery and Data Mining, and it used the TCP dump data provided by the 1998 DARPA ID Evaluation network. The task was to categorize the connections' records into either normal traffic or one of the following attack types: "DoS," "Probing," or "R2L, and ’U2R’." depending on how they had already been preprocessed.

Despite having a number of problems, the KDDCup-'99 datasets are nevertheless considered by [6] to be a useful benchmarking dataset that can be used to test various intrusion detection strategies. The main driver of the success of ML-based techniques is their ability to combat constantly changing, complex, and varied threats while maintaining a manageable false positive rate of ID. Early on, [7] employed the P-rule and N-rule-derived PNrule approach to determine the presence and nonexistence of the class, respectively. Due to the improvement in detection rates for all other sorts of attacks save those in the U2R category, this has a benefit. [8]

Numerous research projects and studies for NIDS have already been undertaken. Through the use of several machine learning and deep learning algorithms, we attempted to identify network intrusion in this study. For the purpose of calculating the accuracy for our dataset, we attempted to compare these two algorithms in this section. It is possible to identify network intrusion using machine learning and deep learning approaches, which makes it easier for researchers to select the most effective machine learning and deep learning approach.

1.1 Background and Motivation

While cyberattacks against the healthcare industry are, sadly, nothing new, the worrying recent surge in malicious threats by attackers has highlighted the need amongst organizations of every size to take a zero-trust approach to infrastructure defense. Investing in good internal cybersecurity, embracing security by design, and ensuring employees are digitally literate – all these can minimize the cybersecurity vulnerabilities that compromise security.

Between January and April 2020, the share of the German workforce working from home increased from 12% to 26%. As well as expecting to continue enjoying non-stop access to a world of entertainment and information, a large number of people now suddenly needed reliable, secure access to company networks – every day.

And that number isn’t going to go down. Indeed, over the next five years, the number of connected devices is forecast to increase from 30 billion to 75 billion – and the more devices there are in a network, the more pleased the hackers will be. As Dr. Ralf Wintergerst, G+D Group CEO, says, “Corona is a major challenge for digital infrastructures. At the same time, it creates a new dynamic of digitalization. This is why we have to raise awareness of cybersecurity and increase our investments in digital infrastructures now.”[9][10]

Furthermore, the amount of computer malware1 has increased rapidly in recent years; *"from about 333,000 in 2005 to 972,000 in 2006, and 5,490,000 in 2007"* [11]

In 2020, the number of new malware attacks declined for the first time since 2015. However, according to SonicWall's 2022 Cyber Threat Report, this was just a temporary dip, with malware attacks now sitting at 10.4 million per year, roughly where they were back in 2018. SonicWall reported 5.4 billion malware attacks took place in 2021, which sounds bad but actually represents a small decrease from the previous year. We don't have full data for 2022 just yet, but the first six months saw 2.75 billion attacks, and if these numbers hold, we'll end up with roughly the same annual number of attacks.[12]

Even if the computer is not connected to the Internet or any other network, everyone using it is still at danger of intrusion (i.e. through physical access). Anyone can try to access the computer and abuse the system if it is left unattended. However, if the computer is linked to a network, especially the Internet, the issue is much worse. Any user in the globe can remotely access the computer (to a certain extent) and attempt to access private or secret information or launch an attack to stop the system from working properly or completely. The act of hacking into a computer system does not require human intervention. With specially designed software, it might be executed automatically. A well known example of this is the Slammer worm (also known as Sapphire), which performed a global Denial of Service (DoS) attack in 2003. The worm exploited a vulnerability in Microsoft's SQL Server, which allowed it to disable database servers and overload networks [13]

Moore *et al*. refer to Slammer as *"the fastest computer worm in history"*, which infected approximately 75,000 computer systems around the world within 10 minutes. Not only did the Slammer worm restrict the general Internet traffic, it *"caused network outages and unforeseen consequences such as canceled airline flights, interference with elections, and ATM failures*" [13]

A private person may not have much at stake if s/he is targeted by a 'cyber attack', but it is a serious threat to professional companies and government organizations. A survey by the Web Application Security Consortium (2008) revealed that 67% of attacks in 2007 were profit motivated. There are many examples in recent news of cyber attacks. For example, early in 2009,

it was revealed that the US power grid had been infiltrated by an intruder, leaving malware that was capable of shutting down the entire grid [14]

A significant spy network (GhostNet) was revealed later that year [15] GhostNet, which is reported to have been primarily based in China, is said to have compromised more than 1000 systems worldwide, with victims including foreign ministers and embassies. When the Russian military was charged with launching denial-of-service operations against Georgia during the conflict over South Ossetia, there was another incident involving the government that was disclosed in 2008.[16] These instances show how cyberattacks can pose a threat to national security, which led President Barack Obama to establish a national cyber security organization in the USA in May 2009 , which was quickly followed by the UK [17]

There are several mechanisms that can be adopted to increase the security in computer systems. [18]

Consider three levels protection:

Attack prevention: Firewalls, user names and passwords, and user rights.

Attack avoidance: Encryption.

Attack detection: Intrusion detection systems.

Despite adopting mechanisms such as cryptography and protocols to control the communication between computers (and users), it is impossible to prevent all intrusions [19] .Firewalls serve to block and filter certain types of data or services from users on a host computer or a network of computers, aiming to stop some potential misuse by enforcing restrictions. However, firewalls are unable to handle any form of misuse occurring within the network or on a host computer. Furthermore, intrusions can occur in traffic that appears normal [20] Intrusion Detection Systems (IDSs) do not replace the other security mechanisms, but compliment them by attempting to detect when malicious behavior occurs.

In general, the goal of an intrusion detection system (IDS) is to identify situations where a user's actions conflict with the intended use of a computer or computer network, such as when they engage in fraud, break into the system to steal data, launch an attack to cause the system to

malfunction or even crash. System administrators manually analyzed user behavior logs and system notifications to perform intrusion detection prior to the 1990s, but their odds of spotting active intrusions were low [21]As software to automatically analyze the data for system administrators was developed, this gradually altered thanks to early research by Anderson (1980) and Denning (1987). Early in the 1990s, the first IDS was created that accomplished this in real-time [21]. The volume of data in modern computer networks, however, makes this a serious difficulty despite the rise in computer use.

IDSs now use a wide variety of Artificial Intelligence (AI) approaches, Rule Based Systems (RBSs) were the first to be successfully used in the beginning and are still the foundation of many IDSs. This makes it possible for IDSs to detect known intrusion patterns by automatically filtering network traffic and/or analyzing user data. The rules that resulted in the intrusion alert can be explained in length and with specificity when reporting suspected intrusions to an administrator. RBSs generally have the disadvantage of being rigid (due to the strict regulations), which prevents them from detecting new incursions or changes of old intrusions. [22]

Machine learning and data mining, which is another branch of AI, provides the needed flexibility and has been the subject of extensive research over the past ten years. Examples of these approaches include artificial neural networks and clustering.

These methods are frequently used as classifiers that learn to automatically detect intrusions from a training set of user behavior or network traffic examples. As a result, there is no need to take information from a human expert and turn it into rules that can describe assaults. The ability of machine learning approaches to generalize from known attacks to variations of those attacks or even detect completely new types of intrusion is a benefit.

The kind of misuse detection that has been mentioned thus far involves an IDS scanning for known (or learnt) threats. Anomaly detection is a different type of detection that is possible with machine learning. All unidentified behavior is viewed as a potential breach since machine learning techniques are able to learn what constitutes typical behavior. Because of this, such systems are able to identify completely new attack types. An rise in false warnings, though, is the price you pay (false positives) [23]

It is difficult to design networks and systems that are safe enough for everyday usage as the world shifts toward becoming gradually more dependent on computers and automation. The expansion of online marketplaces and services has drastically increased the amount of security dangers to enterprises. There are many ways to deal with threats to network security. In order to counteract security risks, intrusion detection systems are installed alongside firewalls in networks. They search the network for every incoming and outgoing traffic, and they examine each packet's signature to determine whether it is malicious or not. The system uses machine learning to profile common network packets and learn the signature of known assaults. [24].

Some of the best intrusion detection systems on the market are:

- Snort
- Bro
- OSSEC
- Suricata
- Sagan
- Security Onion
- Samhain

Recent research focuses more on the hybridization of techniques to improve the detection rates of machine learning classifiers. For example, Sabhnani and Serpen (2003) examine the performance of 9 machine learning algorithms on a commonly used data set, the KDD Cup ’99 data set [25]

First, they discovered that certain approaches worked better on various infiltration types. Second, they discovered that the detector performed better overall when the top methods from each class were combined. However, there are differences between the results that have been reported in the literature about how well various approaches work against various classes of intrusion.

The KDD Cup '99 data set. Contradictory results have been published in the literature, despite the fact that several researchers use the same machine learning techniques on the data set. Additionally, several academics have criticized the data set in publications2, calling into question the veracity of the conclusions drawn using this data [26] Researchers still utilize the data despite the objections because there aren't any better publically available alternatives. As a result, it's

critical to recognize the significance of the data collection and the conclusions drawn from the vast body of research that was used to generate it, both of which have largely been overlooked by the existing criticisms.

Here, it is hypothesized that ANNs can learn from unbalanced data with better preparation. In this thesis, machine learning and multilayer perceptrons —which are frequently trained by deep neural networks—are focused. The classifier's error is to be minimized by this training algorithm. The minor class(es) can therefore be disregarded because the method can obtain a very low error by only correctly categorizing the large class(es). The development of an intrusion detection system with a 100% success rate is challenging or nearly impossible. Many security issues exist in the majority of systems nowadays. Not all incursions are known to exist. Additionally, by leveraging machine learning techniques, hackers are developing new methods of breaking into networks.

The quick identification of these attacks will help in finding potential intrusions. limit the amount of damage done. So, creating an accurate and effective intrusion detection system will assist in lowering network security risks.

1.3 Research questions and constraints

As the research presented in this thesis developed, it naturally formed three main parts: analysis of the KDD cup ’99 data set, Observed different type of attack based on machine learning algorithm, proposed a DNN structure for these dataset. Specific aims and objectives related to each of these parts of the thesis are presents in their respective chapters, whilst general research questions are presented here at a high level.

• What has caused the contradictory findings with the KDD Cup ’99 data set reported in the literature?

• In light of criticisms of the KDD Cup ’99 data set in the literature, can it be used in the future to give valuable contributions to the intrusion detection and machine learning domains?
• Is the poor detection of some classes of intrusion due to issues with learning from imbalanced data?
• How can one utilize machine learning to better learn from imbalanced data?
• Can classifier combination be adopted to better learn from imbalanced data?
• How does the selection of base classifiers affect the performance of the resultant ensemble?

The scope of this thesis has been determined by a number of pragmatic constraints, which have been applied to ensure a focused investigation without compromising the ability to answer the research questions. This is necessary since the thesis considers several large research domains. The majority of the constraints apply to intrusion detection. First, the empirical work in this thesis only considers network based misuse detection. However, the review of the domain considers all the main methods of intrusion detection. Although some challenges and concepts apply to fraud detection and fault localization, these applications are excluded. Furthermore, since the focus of this investigation is on machine learning, other, conventional, techniques applied to intrusion detection are not considered. However, the review does consider a broad range of other AI techniques applied to intrusion detection.

There are several aspects of intrusion detection that are not considered here, although they would require attention when developing an IDS to be deployed in real life, such as:

Architecture: the focus here is on what could be referred to as a detection module that would exist in a larger IDS framework. Especially in wireless and mobile *ad hoc* networks, the architecture is very important. This includes determining where to deploy the IDS, which is considered a general challenge [23]

Data collection: since the KDD Cup ’99 data set is adopted in this work, data collection is not required. It would, however, be necessary to collect data from the environment in which an IDS is to be employed. This also includes a process of labeling data for supervised learning.

Data preprocessing: some data preprocessing is necessary in this work, mainly for the MLPs that are adopted, enumerating and scaling feature values. However, the KDD Cup ’99 data set has already undergone an initial preprocessing task by transforming the raw *tcpdump* from the DARPA data into a feature set suitable for machine learning. Although the availability of this data set is very convenientfor researchers, criticism in the literature indicates that the transformation was not ideal [26]

Performance: there are several mechanisms that can be adopted to help achieve a better performing IDS, in terms of detection rates, speed and memory usage, *e.g.*, feature selection and data sampling. Related to the data transformation discussed above, different transformations and feature sets may facilitate improved intrusion detection. There are also different ways of preprocessing the data for the MLPs adopted in this study, which may be considered better and could yield improved performance. However, the focus of this thesis is on the issues and challenges posed by the research questions, not to develop an optimal IDS prototype.

Other pragmatic considerations: detecting new intrusions will always be a challenge; there will always be new software, which inevitably have vulnerabilities that can be exploited.

Therefore, re-training is necessary once new data is available. When and how this is done is not considered here. Neither is online training or unsupervised learning.

1.4 Thesis organizations

There are eight chapters throughout all of our work on the thesis. Our introduction, history, research tenets, inspiration, and thesis organization are all presented in the first chapter. The literature review is covered in the second chapter, where we also give a brief overview of earlier research that are similar to our own. We introduced the fundamentals of network intrusion detection in the third chapter. We covered the implemented machine learning and deep learning algorithms in our fourth and fifth chapters. We have presented our datasets in the sixth chapter in terms of multiclass classification using our attack classifier. Our study methodology, research findings, and analysis based on machine learning and deep learning algorithms, as well as a general description of machine and deep learning, are then offered in our seventh chapter. Finally, we have provided the summary of our entire thesis work in chapter eight.

Chapter Two

Literature Review

The research on ID in network security has existed since the birth of the computer architectures. The use of ML techniques and solutions to holistic IDS has become common in recent days, but training data at hand is limited and are mostly used only for bench-marking purposes. DARPA datasets [27], are one of the most comprehensive datasets available publicly. The data of tcpdump offered by the 1998 DARPA ID Evaluation network of 1998 was cleaned and utilized for the KDDCup contest of 1999 at the 5th International Conference on Knowledge Discovery and Data Mining. The job was to organize the records of the connections that are already preprocessed into either traffic which is normal, or one of the following categories of attack: 'DoS', 'Probing', 'R2L' and 'U2R'.

The preprocessing of the KDDCup-'99' competition's data was done using the MADAMID framework [28]. The entries that used variants of decision trees showed only marginal differences in performance occupied the first three places [29, 30, 31]. The first 17 submissions of the competition were all benchmarked to perform well and are summarized [32]. The majority of published results were tested and trained with only 10% training set observing the feature reduction on the KDDCup- '99' datasets [33, 34, 35]. Few researchers used custom built datasets, with extracted from the 10% KDDCup-'99' training set [36, 37, 38].

There are a number of interesting publications where the results are indirectly compared due to the use of different training and test datasets. In a paper [39], genetic algorithm and decision trees were used for automatic rule generation for an intelligent system for enhancing the capability of an existing IDS. The integrated utilization of neural networks in IDS was suggested by [40] and [41]. [42] proposed an application of recurrent neural networks (RNNs) and [43] compared the neural network architectures' performance for statistical anomaly detection to datasets from four different scenarios.

Although the datasets of KDDCup-'99' has various issues [44], [45] argues that they are still an effective bench-marking dataset which is publicly available to compare different intrusion detection methods.

The fundamental reason for the popularity of ML-based approaches is because of its capability to attack the constantly evolving complex and diverse threats to achieve an acceptable false positive rate of ID with the reasonable computational cost. In early stages, [46] used PNrule method which is derived from P-rules and N-rules to figure out the existence and nonexistence of the class respectively. This has an advantage due to the enhancement of the detection rate in the other types of attacks except for the U2R category.

An extrapolation to traditional Feed Forward Networks (FFN) in the plane of taking inspiration from biological elements, is a network named Convolutional Neural Network (CNN). In early stages, CNN was used for processing of images by making use of normal 2D layers, pooling 2D layers and completely connected layers. [47] studied the applications of CNN for IDS with the KDDCup of '99' dataset and compared the results with several other bleeding-edge algorithms. After a broad analysis, they have concluded the superiority of CNN over the other algorithms. The study of the utilization of the Long Short-Term Memory (LSTM) classifier was conducted by [48] with the same dataset. It has been stated that because of the capability of LSTM to see into the past and relate the successive records of connections demonstrates usefulness towards intrusion detection systems.

The ultimate motive of this paper is to exploit the possibility of randomness of the inbound cyber attack which is unsuspicious to human sight but can be filtered by adding an artificial intelligence layer to the network. Hence, by training the neural network with the existing cyber attacks data, it can learn to predict an inbound attack easily and can either alert the system or initiate a pre-programmed response which may abstain the attack from proceeding further. As a result, millions worth, aftershock collateral damage and expensive data leaks can be prevented just by simply adding an extra layer to the security system. The benchmarking dataset used for training the networks are bygone and for a better real-time robustness of the algorithm, more recent data must be used for retraining before deploying in the field. The obligatory of this paper is to introduce the essence of artificial neural networks into the much rapidly evolving field of cybersecurity.

Chapter Three

Intrusion detection

In this section we discuss overall infrastructure of intrusion detection system.

3.1 Definitions and terminology

Intrusion detection is the process of monitoring and analyzing events that occur in a computer or networked computer system to detect behavior of users that conflict with the intended use of the system. An Intrusion Detection System (IDS) employs techniques for modelling and recognizing intrusive behavior in a computer system. When referring to the performance of IDSs, the following terms are often used when discussing their capabilities:

True positive (TP): classifying an intrusion as an intrusion. The true positive rate is synonymous with detection rate, sensitivity and recall, which are other terms often used in the literature.
False positive (FP): incorrectly classifying normal data as an intrusion. Also known as a *false alarm*.
True negative (TN): correctly classifying normal data as normal. The true negative rate is also referred to as specificity.
False negative (FN): incorrectly classifying an intrusion as normal. The performance metrics calculated from these are:

$$True\ positive\ rate\ (TPR) = \frac{TP}{TP+FN} = \frac{\#correct\ intrusions}{\#intrusions}$$

$$False\ positive\ rate\ (FPR) = \frac{FP}{TN+FP} = \frac{\#normal\ as\ intrusions}{\#normal}$$

$$True\ negative\ rate\ (TNR) = \frac{TN}{TN+FP} = \frac{\#correct\ normal}{\#normal}$$

$$False\ negative\ rate\ (FNR) = \frac{FN}{TP+FN} = \frac{\#intrusions\ as\ normal}{\#intrusions}$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} = \frac{\#correct\ classifications}{\#all\ instances}$$

$$Precision = \frac{TP}{TP+FP} = \frac{\#correct\ intrusions}{\#instances\ classified\ as\ intrusion}$$

Accuracy is also referred to as an *overall* classification rate, and according to [49] precision is also referred to as *recall*. Due to the direct nature of many intrusions, the terms ‘intrusion’ and ‘attack’ are used interchangeably.

3.2 IDS: CONCEPT AND CLASSIFICATION

This section first explains the concept of IDS and then provides the details about the classification of IDS based on its deployment and the detection methodology

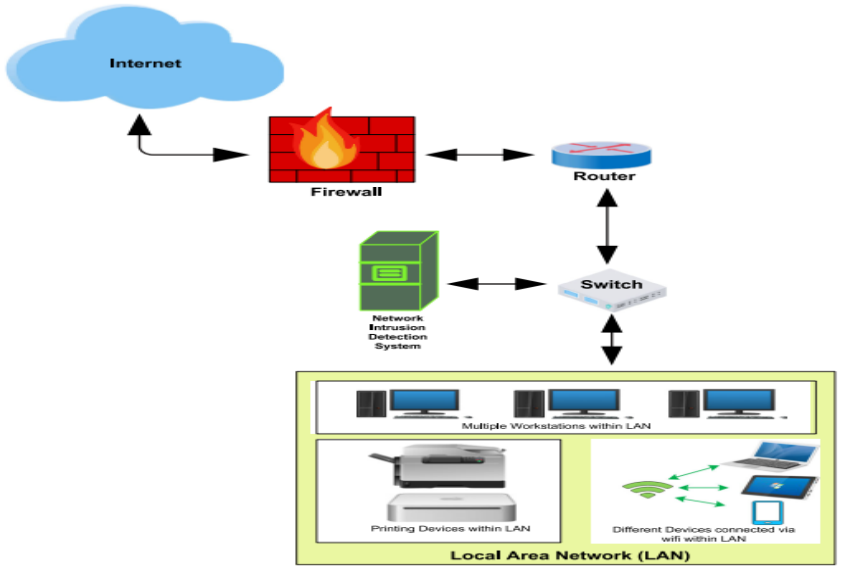

Figure 3.1: CONCEPT AND CLASSIFICATION

3.2.1 Concept

An IDS is the combination of two words “intrusion” and “detection system.” *Intrusion* refers to an unauthorized access to the information within a computer or network systems to compromise its integrity, confidentiality, or availability.

While a detection system is a security tool for identifying such unlawful activities. IDS is a security technology that continuously scans host and network traffic for suspicious activity that violates security policies and directly impacts their confidentiality, integrity, and availability. IDS will send server or network administrators notifications about any detected malicious activities. A passive deployment of NIDS is shown in Figure 3 where it is connected to a network switch that is set up with port mirroring technology. The assignment is to mirror all incoming and outgoing network traffic to NIDS for traffic monitoring and intrusion detection. To allow all traffic to pass through NIDS, NIDS can also be installed between the firewall and the network switch.

3.2.2 Classification of IDS

IDS can be classified with the perspective of its deployment or detection methods. A classification taxonomy is given in Figure 4

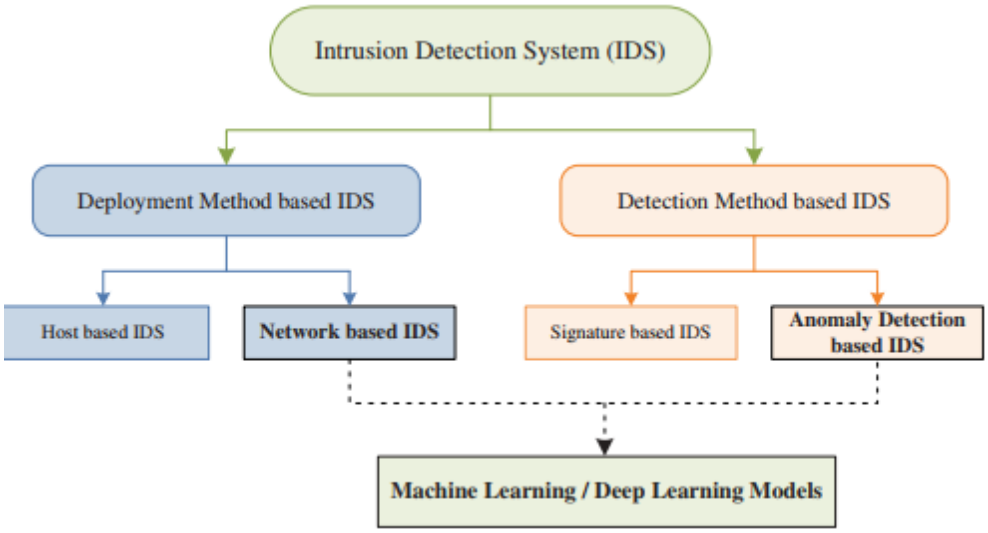

Figure 3.2: Classification of IDS

3.3 Deployment method based IDS

IDS is further classifed as host-based-IDS (HIDS) or NIDS from the deployment-based IDS perspective. On the single information host, HIDS is installed. Its responsibility is to keep an eye on all activity on this one host and scan for any infractions of security guidelines or suspicious activity. The biggest disadvantage is that it must be installed on every host that needs intrusion protection, which adds extra processing overhead to every node and lowers the IDS's overall performance. [50][51] NIDS, on the other hand, are installed on the network with the intention of defending every device and the entire network from intrusions. The network traffic will be continuously monitored by the NIDS, which will search for any potential security infractions. The many approaches utilized in the NIDS are the main topic of this essay.

3.4 Detection method based IDS

Anomaly detection-based intrusion detection (AIDS) and signature-based intrusion detection (SIDS) are further divisions of the IDS from the perspective of detection. The concept of creating a signature for attack patterns is the foundation of SIDS, often referred to as "misuse intrusion detection" or "knowledge-based intrusion detection." The data patterns are compared with these stored signatures for attack detection, and these signatures are kept in the signature database.[52][53]

Due to the availability of attack signatures, the advantage includes the high detection efficiency for known attacks. However, because there are no signature patterns for this approach, it cannot identify fresh or new attacks. [54] It is a resource-intensive approach since a sizable signature database is also kept and examined with the data packets for potential intrusions. [55] The concept of AIDS, also known as the "behavior-based IDS," is to precisely define a profile for typical behavior. Any departure from this expected profile will be viewed as an abnormality or an aberrant action. [56],[57] The primary benefits of AIDS are its capacity to identify novel and unknown attacks[58] and the tailored nature of the typical activity profile for various networks.

The large FAR is the biggest disadvantage, though, as it makes it impossible to distinguish between normal and aberrant profiles for intrusion detection. [59] The use of IoT devices has increased exponentially as a result of the IoT paradigm's acceptance and network technology advancement. [60] The WSN, which consists of a group of sensor nodes for information gathering, is a key technology utilized in the building of an IoT network. These IoT sensor devices gather and disseminate a significant amount of vital information online. [61] The IoT network has security difficulties as a result of this massive data and the complicated structure of WSN, which is made up of resource-constrained sensor nodes. [62],[63] In this regard, IDS is seen as one of the effective mechanism for both IOT and WSN .

The literature has several different IDS strategies that are based on the effective application of watchdogs, trust models, and game-theoretic ideas. The network nodes designated as watchdogs have the duty of keeping an eye on and keeping track of the network traffic of the nearby nodes. The problematic nodes are then decided upon using a set of rules. In the fields of WSN, AdHoc networks, and IoT, numerous solutions for anomaly and intrusion detection employing watchdogs are put forth.[64]

Another approach for enhancing an IDS's performance is trust models. By constantly monitoring the network traffic for unusual behaviors, an IDS based on the trust model assesses the nodes' trustworthiness to detect malicious nodes. Watchdog, Bayesian, and game theory-based trust models are some of the trust models used in various IDS implementations. [65,66] To lessen the computational burden on sensor nodes with limited resources in the IoT, a distributed trust management technique can be implemented. [67,68] In a similar vein, game theory is frequently utilized in the effective design of IDS.

It is an applied mathematical idea that is used to describe a game and model the tactical interactions between players. Every game has a certain number of players, and each player has a unique set of strategies, an action plan, and a payment for every decision they make. The game's solution is based on a stable state that is based on the player's approach to maximizing the reward. Depending on how cooperatively or competitively entities interact, a game might be cooperative or noncooperative. According to IDS for IoT and WSN, a game between attackers and defenders is modeled either by their interaction or by employing an attacker's prediction approach.[69,70]

In this report , we have focused on reviewing the AI-based NIDS, which can be deployed for the security of an IoT network by monitoring the network traffic entered through the edge router.

The most common AI-based algorithms used in the design of an efficient NIDS over the past three years.

3. 5 Intrusion

In general, any behavior that differs from the typical, expected use of the system might be categorized as intrusive behavior. Many of the difficulties faced by fraud detection, fault management, and localization are also faced by intrusion detection. There is a natural overlap between various areas, even if these are not taken into account here, particularly for event correlation. Given the variety of intrusions, it is challenging to provide a single definition of the term.[71]

Surveillance/probing stage: By looking for software and configuration flaws that can be exploited, the intrusive party tries to learn more about possible target machines. Cracking passwords is part of this.

Activity (exploitation) stage: Once bugs have been found in the earlier phase, they can be used to gain administrator rights to the chosen host (s). The hacker will have unrestricted access to the system as a result. Denial of Service (DoS) assaults, which are described in more depth below, may also occur during this period.

Mark stage: After the exploitation step, the attacker may be free to steal data from the system, discard data (including logs that would indicate an attack occurred), install malware such as a virus or spyware, or use the host as a base for carrying out other assaults. then, at this point, the attacker has accomplished his or her attack's primary goal(s). [71]

Masquerading stage: In the last phase, the attacker will try to hide their activity by, for instance, erasing log entries that show the intrusion.The two first steps are subsequently developed into a commonly used attack taxonomy in the literature that classifies attacks while evaluating IDSs and takes into four types of intrusion. [72]

Probing (surveillance): Same as the first stage above.

Denial of Service (DoS): The general purpose of *DoS* attacks is to interrupt some service on a host to prevent it from dealing with certain requests. This may be a step in a multi-stage attack, such as the *Mitnick* attack . Here we describe three types of *DoS* attacks, those that (1) *"abuse legitimate features"*, (2) *"create malformed packets that confuse the TCP/IP stack of the machine that is trying to reconstruct the packet"*, or (3) *"take advantage of bugs in a particular network daemon"*.

User to Root (U2R): These attacks exploit vulnerabilities in operating systems and software to obtain root (administrator) access to the system. For example, the buffer overflow attack: "*Buffer overflows occur when a program copies too much data into a static buffer without checking to make sure that the data will fit. For example, if a program expects the user to input the user's first name, the programmer must decide how many characters that first name buffer will require.*

Assume the program allocates 20 characters for the first name buffer. Now, suppose the user's first name has 35 characters. The last 15 characters will overflow the name buffer. When this overflow occurs, the last 15 characters are placed on the stack, overwriting the next set of instructions that was to be executed. By carefully manipulating the data that overflows onto the stack, an attacker can cause arbitrary commands to be executed by the operating system."

Remote to Local (R2L): There are some similarities between this class of intrusion and *U2R*, as similar attacks may be carried out. However, in this case, the intruder does not have an account on the host and attempts to obtain local access across a network connection. To achieve this, the intruder can execute buffer overflow attacks, exploit misconfigurations in security policies or engage in social engineering.

The four classes above may be used in an IDS for classifying intrusions, rather than only differentiating between 'normal' and 'intrusion'. This gives more information about the type of intrusion, which may affect the chosen method of reporting and acting on the suspected detection. Single events can signify an intrusion, whilst other events are not considered an intrusion before they are observed in the context of one or more other events. This could be a repetition of the same event, as would be typical for *Probing* or *DoS* attacks, or a completely different event. An IDS should be able to recognise simple, single event, attacks as well as complex, multiple event, attacks [73]. As an example of the former Benferhat *et al*. (2003) mention *ping of death*, which is a *DoS* attack where the attacker sends a too large ping package to a host, which may cause it to crash. As an example of the latter, they describe the *Mitnick* attack, which consists of the following steps:

1. An intruder floods the login port a host computer *H* so that it cannot respond to any other requests.
2. The intruder uses *H*'s IP address to send spoofed messages to a server *S*. *S* returns messages to *H*, and normally *H* would return messages to close the connection. However, since *H* is unable to respond, the connection remains open.
3. After being able to open the connection to S, the intruder can attempt further ingress to exploit the system. Performing intrusion detection in wireless, mobile *ad hoc* and sensor networks have become more in focus in recent years. These networks pose additional challenges to IDSs, due to their distributed nature and *ad hoc* infrastructure [74].Additional security threats and operational considerations are necessary to take into account when deploying an IDS in such networks.

3.6 Intrusion detection systems

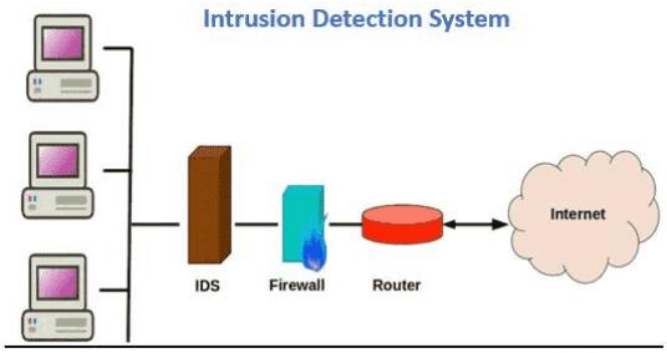

Figure 3.6: Intrusion detection systems

The specific architectures of IDSs are not discussed here, as these are diverse and continue to evolve with time. In general terms, the following common building blocks of an IDS:

Sensor probes: gather data from the system under inspection.

Monitor: receives events from a number of sensors and forwards suspicious content to a 'resolver'.

Resolver: determines a suitable response to suspicious content.

Controller: provides administrative functions. This section focuses on characteristics of IDSs, which elaborate on the first three points above. An IDS may be described according to four characteristics.

Audit source location: host based or network based.

Detection method: misuse or anomaly detection.

Behaviour on detection: passive or active.

Usage frequency: real-time or off-line. Also considered here is 'detection approach', which describes, at a lower level, the strategies used to detect intrusions. This is related to 'detection method', and is discussed to some degree in within misuse detection systems. However, the

detection approach is not constrained to misuse detection, and, thus, is treated as a separate characteristic here. The five characteristics are discussed in their respective sections below.

3.7 Detection method

There are two main detection methods, referred to as misuse detection and anomaly detection. These terms are also known as knowledge based and behaviour based intrusion detection .The former attempts to encode knowledge of known intrusions (misuses), typically as rules, and use this to screen events (also known as a signature based IDS). The latter attempts to 'learn' the features of event patterns that constitute normal behaviour, and, by observing patterns that deviate from established norms, detect when an intrusion has occurred. Some IDSs offer both capabilities, typically via a hybridization of techniques. However, a system may also be modelled according to both normal and intrusive data, which has become a common approach in recent research adopting machine learning techniques . Misuse detection is successful in commercial intrusion detection, *all commercial IDS products were based on misuse detection"*. However, this approach cannot detect attacks for which it has not been programmed, and, thus, it is prone to issue false negatives if the system is not kept up to date with the latest intrusions . On the other hand, misuse detection systems generally produce few false positives

The general perception about misuse detection, as presented above, is no longer entirely accurate. In recent years, researchers have incorporated techniques that allow misuse detection systems to be more flexible, being capable of detecting more variations of attacks. This has been made possible with machine learning techniques such as Artificial Neural Networks, which are built to be able to generalise their models of known attacks to classify unseen cases. This is also the case for rule based systems.

Consider several levels of data sources to model anomaly detection on, which they classify as follows:

Keyboard level: which key is pressed, when the last press occurred, etc.

Command level: Which commands are used and in what order? Researchers now additionally take system call arguments and output parameters into consideration.

Session level: monitoring session conclusion events, which can yield information like "duration of session, overall CPU, memory and input-output utilization, name of terminal utilized, time of login, day of week. note that as the data is only collected after the user has finished a session, by the time the user may have finished the intrusion, this approach is unlikely to be able to conduct real-time intrusion detection.

Group level: assembling users into teams.

An anomaly detection system (also known as user profiling) may create multiple user profiles based on any of these levels. This can be done by either taking into account one profile per user or, at the higher level, groups of users who might have certain system rights, such as administrators, programmers, secretaries, etc. Keeping up with environmental changes is a difficulty for host-based anomaly detection systems. To prevent a rise in false alarms, also known as behavioral drift, retraining or ongoing updating is necessary.

It is feasible to model or train an anomalous system over time, but there is a risk that the system will also learn intrusive behavior When a user is aware that an anomaly detection system is being trained, they may gradually alter their behavior to prevent a planned attack from being discovered.

3.7.1 Detection approaches

Here we discuss stateful and stateless intrusion detection techniques. Stateless approaches try to categorize individual occurrences as an incursion or not, whereas stateful approaches view an assault as being made of numerous events (stages).For the sake of simplicity, stateful approaches—which have been the subject of extensive research and are frequently used in commercial IDSs—such as EMERALD eXpert and eXpert-BSM, HP OpenView , Snort , and a well-known open source IDS called Snort are all considered synonymous with event correlation .Following a discussion of the advantages and disadvantages of the two strategies, event correlation and stateless intrusion detection are further examined in their respective sections below..

3.8 Overview of Intrusion Detection Systems

A complete security solution against network attacks has three main security components: attack prevention, attack detection, and attack reaction [75]. These components are presented in Figure 3.8.

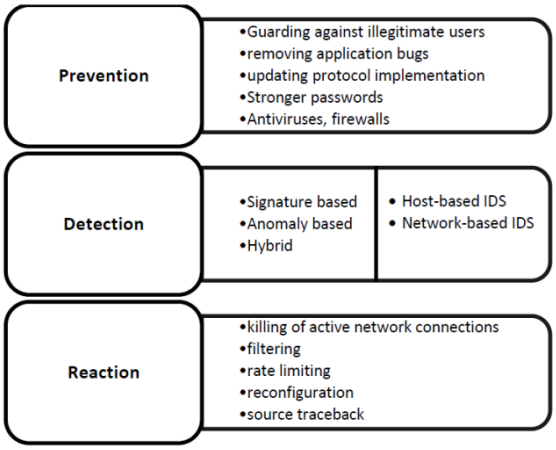

Figure 3.8: Overview of Intrusion Detection Systems

Prevention - Through the installation of necessary security devices, the elimination of application defects, the upgrading of protocol implementation, and the enhancement of the security of all Internet-connected machines, the preventative phase seeks to boost the system's overall security. The network administrator employs preventive measures in this stage to try and protect the system from unauthorized users. The objective is to make it more difficult for attackers to conduct denial-of-service attacks, even though this cannot be done for all attacks.[76]

Detection - An effective attack detection phase should come first in an effective defense system. Each attack detection system's objective is to find intrusions before they cause any significant harm. Any unauthorized attempt to obtain, alter, or destroy data with the goal of rendering a system unreliable is referred to as an intrusion . Detecting attacks quickly is possible with a decent system.

A hybrid system that can utilize the benefits of both signature-based and anomaly-based IDSs offers more thorough defense against a variety of threats.Each category involves a variety of approaches, including threshold detection, statistical measurement, rule-based approaches, and evolutionary computation techniques. [78]

For example in a threshold detection approach, the user specifies threshold values for network traffic and any deviation from the values is considered an attack and the system produces an alarm. This approach is a very common technique used in sensor networks. The detection system recognises an event of interest, when the sensory readings exceed a predefined threshold value [79]

Similarly, a statistical technique determines a network's normal traffic distributions and if these distributions change significantly, the system triggers an alarm. In contrast, rule-based systems come with predetermined sets of rules. When a match is made between an input record and a rule in the system, a rule-based classifier is triggered. Lastly, the use of evolutionary computation approaches becomes a crucial technology in many research endeavors in this field, but enhancements are probably conceivable. To learn about normal and abnormal behavior, these learning and adaptive strategies have frequently been combined with rule-based approaches. [80]

Reaction:

Artificial neural networks (ANNs), fuzzy logic (FL), genetic algorithms (GAs), genetic programming (GP), swarm intelligence (SI) and artificial immune systems (AISs) are examples of computational intelligence approaches, that have been used for solving intrusion detection problems. Considering the source of data being used for intrusion detection, IDSs are classified into two categories; Host-based IDS (HIDS) and Network-based IDS (NIDS)

A HIDS monitors activities taking place inside a specific computer system while running on a single host. Therefore, HIDS offer security for important computers that could contain private information. However, because all of the computers in the network are secured by this NIDS, NIDSs are not limited to packets that are directed at a particular host. A NIDS tracks and analyzes the traffic in a network segment to look for unusual activity. Correlators are used to prioritize alarms from several different detection systems, combining the benefits of host-based and network-based IDSs while minimizing human engagement.

Correlators are able to prioritize warnings based on the degree of harm to a key resource, cluster alerts that are related, and weigh alerts. By minimizing the amount of information analysts must examine in order to detect assaults, this strategy can be helpful.

CHAPTER FOUR

Introduction to Machine Learning Algorithms

The term "machine learning algorithm" refers to a computing method that uses input data to accomplish a task without being explicitly written (i.e., "hard coded") to do so. These algorithms are "soft programmed" in that they automatically modify or alter their design as a result of repetition (i.e., experience) to become ever better at doing the intended task. The process of adaptation known as "training" entails supplying input data samples together with the desired outcomes. The algorithm then refines itself so that it can both generalize to produce the desired outcome when given the training inputs and can also provide the desired result without the training inputs.

when new, unknowing data is presented, the desired outcome. This training constitutes the "learning" part of machine learning. A single adaptation over a predetermined amount of time does not have to be the only adaptation that is trained for. A good algorithm can learn new things as it processes new data, just like individuals can. Machine learning has applications in deferent area like Natural Language processing[91][92][93][94], medical[95], cyber security[96][97][98][99].

As a result of training, an algorithm might change in many different ways. To achieve the best outcomes, the input data can be selected and weighted. It is possible to modify the algorithm's changeable numerical parameters through iterative optimization. In order to achieve the best results, it can organize a network of potential computational routes. The probability distributions created using the supplied data can be used to predict results.

The goal of machine learning is to emulate how people (and other sentient species) learn to process sensory data as input in order to accomplish a goal. The student would have to distinguish between apples and oranges in order to complete this objective, which may be a pattern recognition test.

We can usually tell an apple from an orange even though they are both unique.A computer can be taught to learn to recognize apples and oranges by repeatedly being exposed to genuine apples and oranges, as opposed to hard-coding a computer with a variety of exact approximations of apples and oranges. This supervised learning example pairs each training example of input data (color, shape, odor, etc.) with its associated predetermined classification label (apple or orange).

It makes it easier for the learner to deal with similarities and differences when the items to be categorised have a variety of modifiable characteristics within their own classes yet still include essential characteristics that distinguish them. The ability to identify an unfamiliar apple or orange is the most crucial test of a capable learner. Another sort of machine learning is what is known as the unsupervised algorithm. Throwing a dart at a target could be the objective. There are several degrees of freedom available to the device (or person) in the mechanism that controls the dart's trajectory. As opposed to attempting to pre-program the kinematics, the pupil practise throwing the dart.

Every trial involves a different adjustment to the kinematic degrees of freedom, causing the dart to approach the target ever-closer. Since the training did not establish a connection between a particular configuration of kinematic input and an output, this is unsupervised learning. The algorithm derives its own route from the training data. It should be possible for the skilled dart thrower to change the target.

Thirdly, there is semi-supervised learning, which involves labeling some data while leaving other data unlabeled. In this instance, using the labeled part will aid the unlabeled part in learning more quickly.

This scenario is closer to how humans learn skills than most natural systems are. **[53]**. We have used only supervised machine learning algorithms regarding fake review detection. The supervised machine learning algorithms that we have used for fake review detection are as follows:

1. Multinomial Naive Bayes
2. Bernoulli Naive Bayes
3. Random Forrest
4. K-Nearest Neighbors
5. Logistic Regression
6. Decision Tree
7. Support Vector Machine
8. Ada boost

4.1 Multinomial Naive Bayes

There are several applications or tools available for evaluating numerical data, but very few for analyzing words. One of the most popular supervised learning classifications for categorical text data analysis is multinomial naive Bayes. [81]-[85].

A common Bayesian learning strategy in Natural Language Processing (NLP) is the Multinomial Naive Bayes method (NLP). The program calculates the tag of a text, such as an email or a newspaper article, using the Bayes theorem. It determines the likelihood of each tag for a particular sample and outputs the tag with the best possibility.

Bayes theorem, formulated by Thomas Bayes, calculates the probability of an event occurring based on the prior knowledge of conditions related to an event. It is based on the following formula:

$P(A|B) = P(A) * P(B|A)/P(B)$ ..... (4.1)

Where we are calculating the probability of class A when predictor B is already provided.

P(B) = prior probability of B

P(A) = prior probability of class A

P(B|A) = occurrence of predictor B given class A probability

This formula helps in calculating the probability of the tags in the text**.**

4.2 Bernoulli Naive Bayes

Bernoulli Naive Bayes belongs to the Naive Bayes family. Only binary data is accepted. The simplest illustration is figuring out whether a word appears in a document for each value.

That is a simple model. In situations when counting the frequency of words is less important, Bernoulli might produce superior results. Alternatively, we must count each value binary term occurrence feature, such as whether a word exists in a document or not. These qualities are used instead of determining how frequently a word appears in the text. In simple terms, the outcomes of the Bernoulli distribution are P(X=1)=p or P(X=0)=1-p, which are mutually exclusive. The BernoulliNB theorem might have many aspects, but each one is expected to be distinct.[82].

$P(xi \mid y) = P(i \mid y)\, xi + (1 - P(i \mid y))(1 - xi)$ ..... (4.2)

According to the decision rule formula, x needs to be binary. Think about the formula in the case where xi=1 and the case where xi=0. So, i is the event where xi=1 or the event where xi=0.

4.3 Random Forrest

In contrast to a random forest, which is made up of categorization trees, a forest is a collection of trees.

A classification tree is a structure that is made with the entities of other dependent variables residing on the intermediate nodes and the members of the class variable residing on the leaf nodes. Class variables, sometimes called decision variables or predictor variables, might include

yes/no choices to anticipate disease or loan acceptance, spam/no spam in emails, good/bad/moderate choices for product qualities, 0-9 choices for handwritten digits in pattern recognition, and more. In order to add a new classification tree to the forest, add it to each of the individual classification trees that are produced by random forests[83].

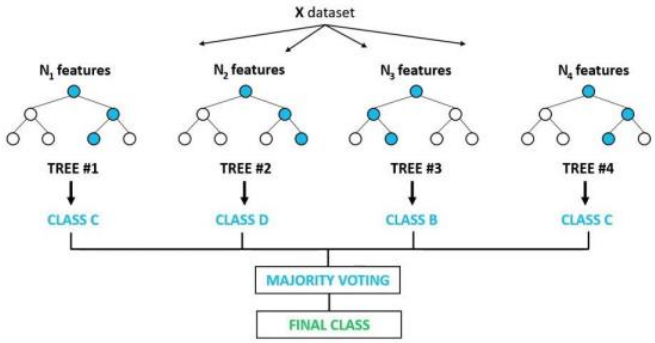

Figure 4.1: Random Forrest classifier

The Random Forest's Construction:
1. If the training set contains N cases, pick all of them at random as a distinct collection of data from the original data set.
2. Choose Tt attributes at random from the training data set for T number of attributes so that the optimal selection of t variables is chosen to split each node. The value of t should remain constant as the tree grows.

3. Without pruning, each tree will develop to its maximum potential. The random forest's error rate is determined by the following two factors:

I. The error rate will rise only if and only if the correlation between any two trees in the forest rises.

II. The error rate determines the tree's strength; the lower the error rate, the stronger the tree, and the forest as a whole.

Random Forest Properties:

• It is often regarded as the most accurate algorithm.

• It is extremely efficient on large data sets, even when there are hundreds of thousands of input variables, and there is no need for data pruning.

• It is particularly efficient when it comes to feature subset selection and missing data imputation.

• During the forest development phase, the random forest algorithm generates an internal unbiased estimate of the generalization error.

• The created forest will be suitable for future data addition .

4.4 K-Nearest Neighbors

One of the simplest machine learning algorithms, based on the supervised learning method, is K-Nearest Neighbor. K-nearest-neighbor (kNN) classification is one of the most fundamental and simple classification techniques when there is little to no prior knowledge about the distribution of the data. It need to be among the first options considered in a classification research. K-nearest neighbor classification was created in order to perform discriminant analysis when accurate parametric estimates of probability densities are either unknown or impossible to obtain.

KNN determines how far a query is from every instance in the data, selects the K examples that are the closest match, and then votes for the label that is used the most frequently (for classification) or averages the labels (for regression)[84].

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we
need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

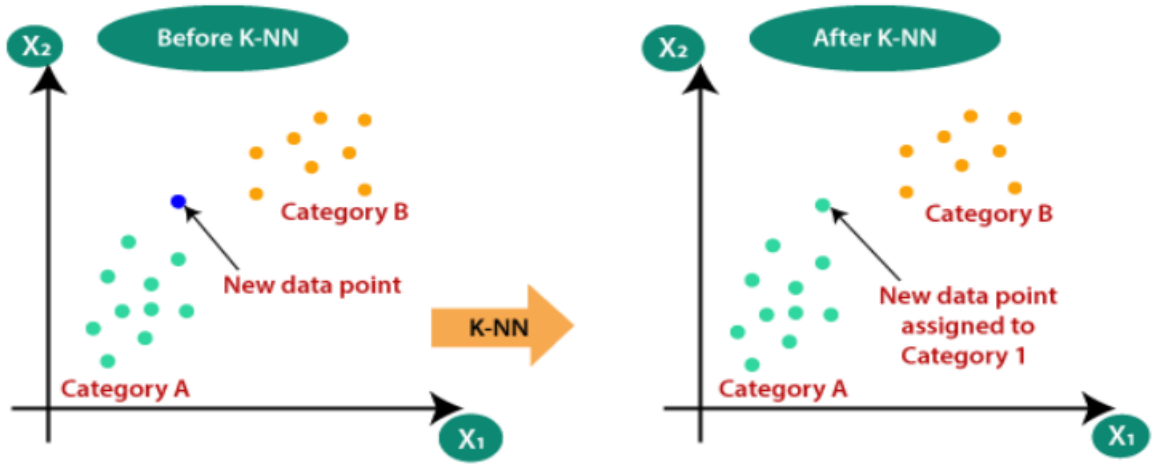

Figure 4.2: K-Nearest Neighbors classifier

Step-1: Select the number K of the neighbors
Step-2: Calculate the Euclidean distance of **K number of neighbors**
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
Step-6: Our model is ready **.**

4.5 Logistic Regression

The method of classification To categorize observations into a set of discrete classes, logistic regression is utilized. Examples of categorization problems include whether an email is spam, whether an online transaction is fake, and whether a tumor is malignant or benign. A probability value is created from the output of logistic regression using the logistic sigmoid function. It is a predictive analytical technique that is founded on the idea of probability[85].

Similar to a linear regression model, a logistic regression uses a more complex cost function known as the sigmoid function or the "logistic function" rather than a linear function. The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

$0 \leq h\theta(x) \geq 1$ ……………………………………………………………………………………….. (4.3)

When using linear regression, we used a formula of the hypothesis i.e.
$h\Theta(x) = \beta_0 + \beta_1 X$ .... (4.4)
For logistic regression we are going to modify it a little bit i.e.
$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$ .... (4.5)
We have expected that our hypothesis will give values between 0 and 1.
$Z = \beta_0 + \beta_1 X$ .... (4.6)
$h\Theta(x) = \text{sigmoid}(Z)$ .... (4.7)
i.e. $h\Theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$ .... (4.8)
This is the basic concept of Logistic Regression **[60]**.

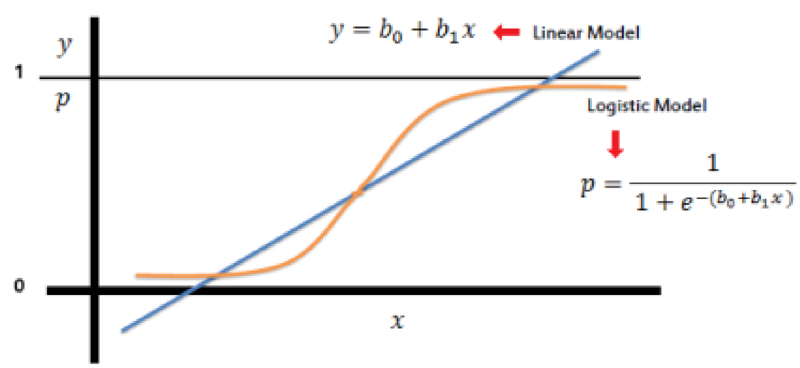

Figure 4.3: Logistic Regression

4.6 Decision Tree

The decision tree is the most effective and popular categorization and prediction tool. Each leaf node (terminal node) carries a class label, whereas each internal node depicts a test on an attribute, each branch a test result.

A tree can be "trained" by breaking the source set up into subgroups depending on attribute value testing. Repeating this approach for each derived subset is known as recursive partitioning.

The recursion ends when all of the subsets at a node have the same value for the target variable or when splitting no longer improves the predictions. Building a decision tree classifier does not require programming knowledge or domain understanding[86].

Using decision trees, instances are sorted from the root of the tree to a leaf node, which offers the classification. As seen in the figure above, an instance is categorised by starting at the tree's root node, checking the attribute that node specifies, and then moving along the tree branch in accordance with the attribute value. The subtree of the new node is then treated similarly.

4.7 Support Vector Machine

SVMs, sometimes referred to as support-vector networks, are supervised learning models that look at data for regression and classification in machine learning.A model created by an SVM training algorithm categorizes fresh instances into one of two categories.Given a set of training examples, each designated as falling into one of two categories, the SVM becomes a non-probabilistic binary linear classifier (although there are ways to apply SVM in a probabilistic classification environment, such as Platt scaling). By using the kernel approach, which entails implicitly translating their inputs into high-dimensional feature spaces, SVMs may do both linear and non-linear classification[87].

Every piece of data is represented as a point in n-dimensional space, where n is the number of features you have and each feature's value corresponds to a certain location in the SVM method. Then, categorization is achieved by identifying the hyper-plane that clearly separates the two groups (look at the below snapshot).

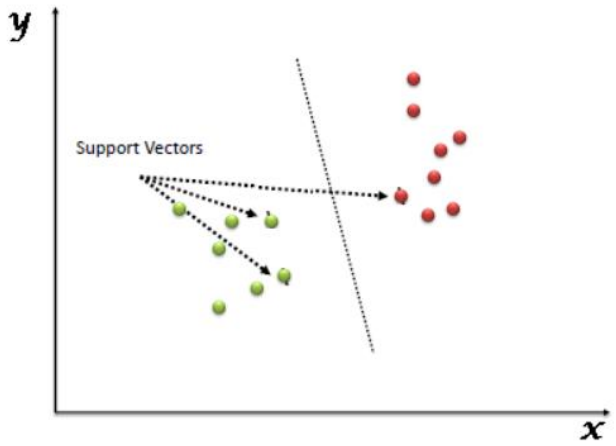

Figure 4.4: Support Vector Machine

Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line) **[63]**.
These are the algorithms that we have used. The result of these algorithms is presented in the Chapter Seven.

4.8 Evaluation Parameters

To understand classifier model's performance, we need to be familiar with some evaluation parameters. A confusion matrix is a table that is used to describe the performance of a classifier algorithm by evaluating the accuracy of it. The elements of confusion matrix are:
True Positive (TP): Which results when classifier model correctly predicts the positive class.
True Negative (TN): Which results when classifier model correctly predicts the negative class.
False Positive (FP): Which results when classifier model incorrectly predicts the positive class.
False Negative (FN): Which results when classifier model incorrectly predicts the negative class.

Table 1.1 : Confusion Matrix

Predicted Values		
Actual Values	Predicted Positive (1)	Predicted Negative (0)
Actual Positive (1)	True Positive (TP)	False Negative (FN)
Actual Negative (0)	False Positive (FP)	True Negative (TN)

Based on the data of confusion matrix, precision, recall, F-measure, and accuracy are the evaluation measures used for evaluating performance of classifier **[64]**.
Precision: Precision is the ratio of correctly predicted positive results to the total predicted positive

results. It measures the exactness of the classifier result **[64]**.

$$Precision = \frac{TP}{TP+FP} \quad \text{..... (4.9)}$$

Recall: Recall measures how accurately classifier model identifies and returns True Positives data. It also refers as True Positives rate. A higher recall is essential for a better classifier model.

$$Recall = \frac{TP}{TP+FN} \quad \text{..... (4.10)}$$

F-measure: F-measure also refers as F-1 score, is the harmonic mean of the precision and recall. It is required to optimize the system towards either precision or recall which have a more influence on final result **[64]**.

$$F\text{-}measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad \text{..... (4.11)}$$

Accuracy: It is the most intuitive performance measure. It can be calculated as the ratio of correctly classified reviews to total number of reviews **[64]**

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad \text{..... (4.12)}$$

CHAPTER FIVE

Introduction to Deep Learning Algorithms

Artificial neural networks are used in deep learning, often referred to as deep structured learning, a sort of machine learning technology, to learn representations. Unsupervised, semi-supervised, and supervised learning are the three different types of learning.

In fields like computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection, and board game programs, deep-learning architectures like deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks, and convolutional neural networks have been used. These architectures have produced results that are comparable to, and in some cases superior to, traditional approaches[88].

The term "deep" in deep learning refers to the network's use of several layers.A network with one hidden layer of infinite width and a nonpolynomial activation function, however, can be a universal classifier, contrary to early research, which holds that a linear perceptron cannot. Deep learning is a more modern variation that uses an unbounded number of layers with bounded sizes, enabling practical application and optimization while upholding theoretical universality in benign circumstances [65].
We have used five dense layer network for our deep learning section

5.1 Simple Dense layer

A neural network with dense layers is one in which each neuron in the layer above is connected to every other neuron in the layer below. This layer is the most well-liked one in artificial neural network networks.

In a model, each neuron in the preceding layer contributes information to the neurons in the dense layer, which also does matrix-vector multiplication. In matrix vector multiplication, the row vector of the output from the earlier layers is the same as the column vector of the dense layer. The row vector and column vector in a matrix-vector multiplication must both have the same number of columns[89].

The general formula for a matrix-vector product is:

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$
$$= \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}. \quad \text{..... (5.1)}$$

Where x is a matrix with a diagonal of 1, and A is a (M x N) matrix. The values under the matrix, which represent the trained parameters of the preceding layers, can be updated through backpropagation. Backpropagation is the most often used algorithm for training feedforward neural networks. Backpropagation calculates the gradient of the loss function for a single input or output in a neural network with regard to the network weights. From the knowledge above, we may infer that the output of the thick layer is an N-dimensional vector. The diameters of the vectors are decreasing, as is evident. In order to use each neuron to adjust the vectors' dimension, a dense layer is used. Every neuron in the layers above delivers its output to every neuron in the dense layer, as was already mentioned. This output then travels through the dense layer, which should contain a count of N neurons, assuming the preceding layer generates a (M x N) matrix by averaging the outcomes of each neuron [90].

CHAPTER SIX

Dataset

6.1 Datasets description:

The DARPA's program for ID evaluation of 1998 was managed and prepared by Lincoln Labs of MIT. The main objective of this is to analyze and conduct research in ID. A standardized dataset was prepared, which included various types of intrusions which imitated a military environment and was made publicly available. The KDD intrusion detection contest's dataset of 1999 was a well-refined version of this .

Using these dataset we divide it into two types of class. One is binary classification and another one is multiclass classification . Binary classification is the task of classifying the elements of a set into two groups (each called class) on the basis of a classification rule. In multiclass classification, each record belongs to one of three or more classes, and the algorithm's goal is to construct a function which, given a new data point, will correctly identify the class into which the new data point falls.

In multiclass classsification we label our attack into five categories.

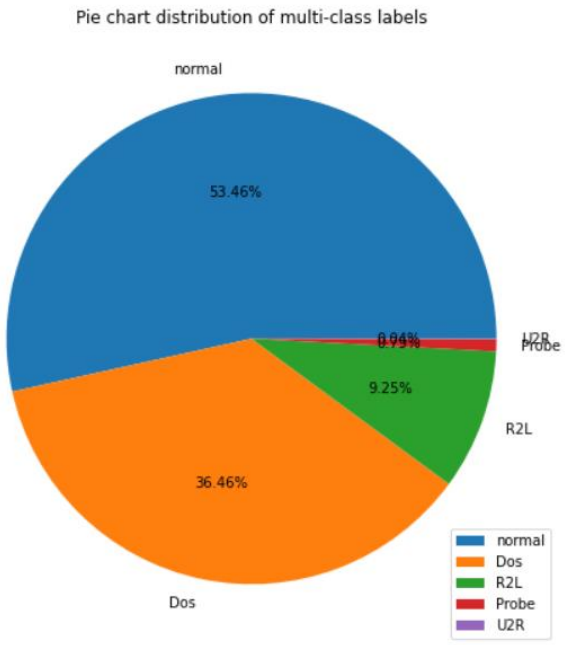

Figure 6.1: multiclass classification

6.1.1 Data preprocessing:

We use those library's for data preprocessing, basically numpy, and pandas.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
%matplotlib inline
```

Figure 6.2: Importing libraries

At first we upload the dataset . As our dataset is in .csv file and already split into two parts one is training part another is testing part. We divide the hole dataset into 70% for training and 30% tor testing.

```
[ ]  train=pd.read_csv('/content/Test.txt',sep=',')
     test=pd.read_csv('/content/Train.txt',sep=',')
```

Figure 6.3 : Reading data file

For preprocessing our dataset, we use pandas and then labelling our dataset into 43 column .

```
[ ] columns=["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","urgent","hot",
     "num_failed_logins","logged_in","num_compromised","root_shell","su_attempted","num_root","num_file_creations",
     "num_shells","num_access_files","num_outbound_cmds","is_host_login","is_guest_login","count","srv_count","serror_rate",
     "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate",
     "dst_host_count","dst_host_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
     "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate",
     "dst_host_srv_rerror_rate","attack","last_flag"]
```

Figure 6.4: Set all the column name

We drop the null values from the dataset.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22543 entries, 0 to 22542
Data columns (total 43 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   duration                     22543 non-null  int64
 1   protocol_type                22543 non-null  object
 2   service                      22543 non-null  object
 3   flag                         22543 non-null  object
```

Figure 6.6: Finding null value

Then we determine count, mean, std, min 20%, 50% , 75%, and max value for each and every attributes .

	count	mean	std	min	25%	50%	75%	max
duration	22543.0	218.868784	1407.207069	0.0	0.00	0.00	0.00	57715.0
src_bytes	22543.0	10395.911369	472796.912692	0.0	0.00	54.00	287.00	62825648.0
dst_bytes	22543.0	2056.110012	21219.763847	0.0	0.00	46.00	601.00	1345927.0
land	22543.0	0.000311	0.017619	0.0	0.00	0.00	0.00	1.0

Figure 6.7: STD report

Then we categories all the data attribute into five kind of attacks both train data and test dataset.

In attack_class normal means 0, DOS means 1, PROBE means 2, R2L means 3 and U2R means 4.

```
[ ] train.loc[train.attack=='normal','attack_class']=0

    train.loc[(train.attack=='back') | (train.attack=='land') | (train.attack=='pod') | (train.attack=='neptune') |
              (train.attack=='smurf') | (train.attack=='teardrop') | (train.attack=='apache2') | (train.attack=='udpstorm') |
              (train.attack=='processtable') | (train.attack=='worm') | (train.attack=='mailbomb'),'attack_class']=1

    train.loc[(train.attack=='satan') | (train.attack=='ipsweep') | (train.attack=='nmap') | (train.attack=='portsweep') |
              (train.attack=='mscan') | (train.attack=='saint'),'attack_class']=2

    train.loc[(train.attack=='guess_passwd') | (train.attack=='ftp_write') | (train.attack=='imap') | (train.attack=='phf') |
              (train.attack=='multihop') | (train.attack=='warezmaster') | (train.attack=='warezclient') | (train.attack=='spy') |
              (train.attack=='xlock') | (train.attack=='xsnoop') | (train.attack=='snmpguess') | (train.attack=='snmpgetattack') |
              (train.attack=='httptunnel') | (train.attack=='sendmail') | (train.attack=='named'),'attack_class']=3

    train.loc[(train.attack=='buffer_overflow') | (train.attack=='loadmodule') | (train.attack=='rootkit') | (train.attack=='perl') |
              (train.attack=='sqlattack') | (train.attack=='xterm') | (train.attack=='ps'),'attack_class']=4
```

Figure 6.8: Categorized all attack class into five classes

6.1.2 Feature Extraction

By generating new features from the current ones, feature extraction attempts to decrease the number of features in a dataset (and then discarding the original features). The majority of the information in the original collection of features should then be summarized by this new, smaller set of features.

Here we use two technique for Feature Extraction:

1. Fill missing value.
2. Compensation of null and missing value.

6.1.3 Data Visualization

Since the year of release of KDD-'99' dataset , it is the most vastly utilized data for evaluating several IDSs. This dataset is grouped together by almost 4,900,000 individual connections which includes a feature count of 43. We describe our dataset through five types of attack and we use KDD-'99' dataset.

At first we see attack class distribution where we could see the highest attack is in zero position is normal attack. Second one is DOS attack , third one is probe attack , fourth one is R2L attack and the last one which is the smallest amount of attack is U2R attack. We could see that after normal attack ,DOS attack is the most in this dataset.

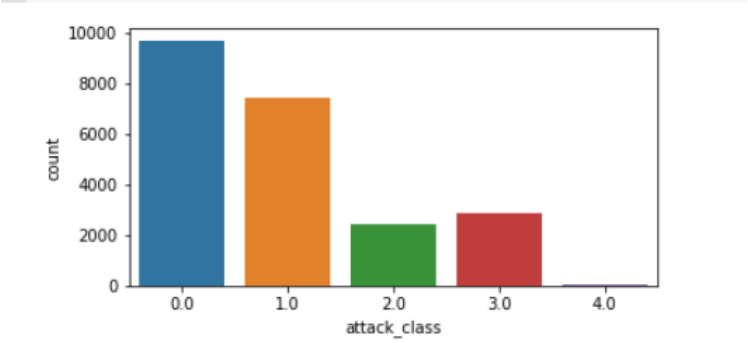

Figure 6.9: Bar chart of attack

There are three types of protocol type distribution . A network protocol is an established set of rules that determine how data is transmitted between different devices in the same network. Essentially, it allows connected devices to communicate with each other, regardless of any differences in their internal processes, structure or design.

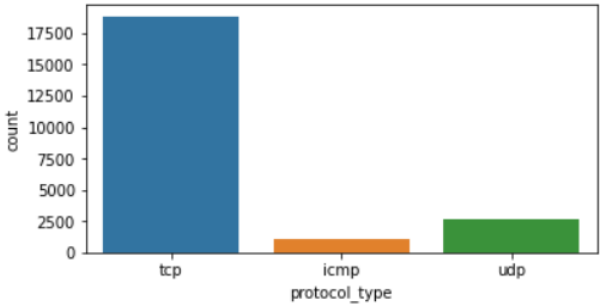

Figure 6.10: Protocol type distribution

Service distribution is another types of distribution from training dataset. From these distribution , we could see http sites are given the highest service among all the services.

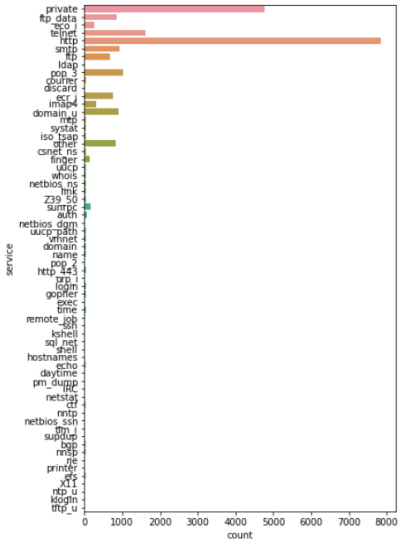

Figure 6.11: Service distribution

Attack distribution : Mainly we have different kinds of attack. But to analyze then in an efficient way we distribute then into 5 classes later , if we see them individually then, we could see normal attack is the most as its ratio is high.

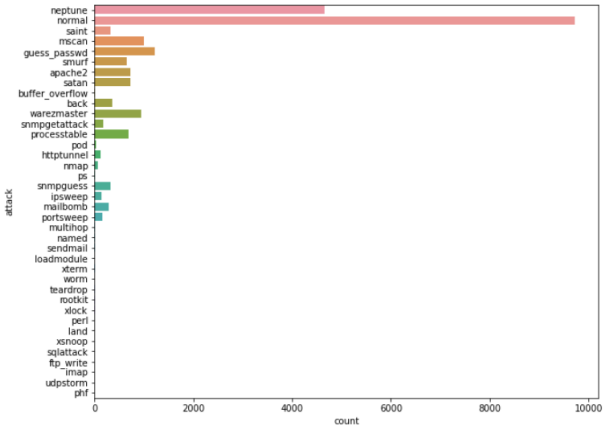

Figure 6.11: Attack distribution

There is no missing in train dataset . So , Missing treatment not required .

	protocol_type	service	flag	attack
N	22543	22543	22543	22543
NMISS	0	0	0	0
ColumnsNames	tcp 18879 udp 2621 icmp 1043 Name...	http 7853 private 4773 telnet 162...	SF 14875 REJ 3849 S0 201...	normal 9711 neptune 465...

Figure 6.12: No missing value

6.1.4 Correlation matrix

A correlation matrix is simply a table which displays the correlation coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values in a table. It is a powerful tool to summarize a large dataset and to identify and visualize patterns in the given data.So this is the correlation structure of KDD-99 dataset, where we could see the correlated line is properly gone through diagonally among all the attributes.

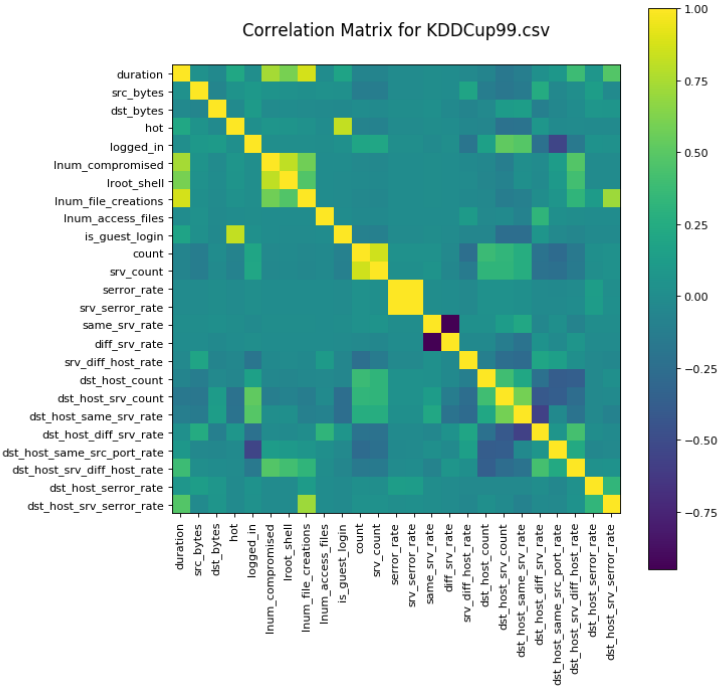

Figure 6.13: Correlation

6.2 Proposed model

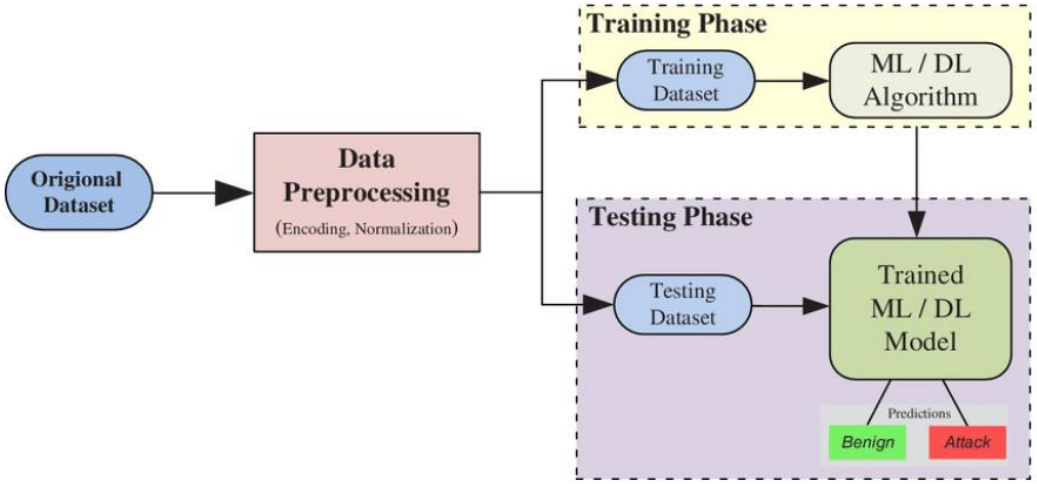

Figure 6.14: Proposed model

CHAPTER SEVEN

7.1 Result and Analysis

For our research work, we have chosen KDDCUP-99 dataset containing different types of attack which are labelled as Normal,DOS,Probe,U2R,R2L. In this paper, we study the correlation matrix among all the features. This dataset is grouped together by almost 4,900,000 individual connections which includes a feature count of 43. In this analysis, we have implemented both machine learning and deep learning algorithms. The machine learning algorithm classifiers are Naive Bayes, Decision Tree, Ada boost, Random Forrest, k-Nearest Neighbors, Logistic Regression, Support Vector Machine (Linear and rbf) to detect attacks using supervised learning. For feature extraction we use dummy variable and variable reduction using Select K-Best technique. For deep learning algorithms, we implemented Deep Neural Network to detect atacks using supervised learning. Our Obtained experiment Results by Using Machine Learning Classifiers are given below.

Table-2

ML Classifier	Accuracy	precision	Recall	F1
Logistic Regression	92.318	98.888	81.998	89.657
Gaussian Naive Bayes	92.983	98.889	92.312	95.534
K-nearest neighbor	92.312	99.898	97.012	98.412
Decision Tree	91.615	99.998	91.612	95.545
AdaBoost	92.067	99.556	91.171	95.321
Random Forest	92.829	99.998	91.038	95.235
SVM-rbf	90.324	98.998	96.985	98.435
SVM-linear	90.246	98.677	96.788	98.412

Here, Deceptive Opinion Spam Hotel Dataset **,** we applied data pre-processing by as applying tokenization, lowercase, stopwords remove, stemming and lemmatization. For feature extraction we applied TF-IDF. We have applied the machine learning algorithms using both unigram and bigram feature. Unigram is a set of continuous words from a given text in which the occurrence of each word is independent of its previous word.

By applying the data pre-processing model, we have achieved test accuracy rate of 92.983% with Gaussian Naïve bayes classifier. The overall precision, recall and F-1 score are also quite high in testing dataset.

In Random Forrest classifier, the test accuracy is 92.829%. It is the best accuracy we get. It comes very high because as it can handle large dataset efficiently .So random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm. Here, we set the number of trees in the forest (n_estimators)=100, the maximum depth of the tree (max_depth)=3, which means using all processors. Test precision, recall and f-1 score of 99.998%, 91.038%, 95.235% which is quite high.

In Logistic Regression, the test accuracy is 92.318%. The reason Logistic Regression has a quite high test accuracy because it is a binary classification and will work best on binary labels. The precision, recall and f-1 score are also quite high in Logistic Regression.

In k-Nearest Neighbors using, we achieve test accuracy of 92.312%. We change n_neighbors = 10, which is the number of neighbors to use to achieve this test accuracy. Its precision,recall,f1 score is good enough.

The Decision Tree classifier shows quite good in test accuracy which is 91%. The precision, recall and f-1 similar to as others classifier.

In both Support Vector Machine (linear) classifier, we have achieved test accuracy almost 90%. The test precision, recall and f-1 score are also good and this indicates how well the classifier has managed to classify each level.

In Ada boost classifier using, we achieve test accuracy of 92.067%.. Its precision,recall,f1 score is good enough. As it a binary classification so it works great on binary label. AdaBoost uses an iterative approach to learn from the mistakes of weak classifiers, and turn them into strong ones.

7.2 Applying Confusion Matrix for Machine Learning Algorithms

Here, we have applied the confusion matrix on our machine learning classifier models (unigram) such as Naive Bayes, Ada boost, Random Forrest, k-Nearest Neighbors, Logistic Regression, Support Vector Machine (Linear) and Decision Tree to observe the performance of the models.

Naïve Bayes

Figure 7.1 : Naïve Bayes confusion matrix

Logistic regression

Figure 7.2 : Logistic regression confusion matrix

K-nearest neighbor :

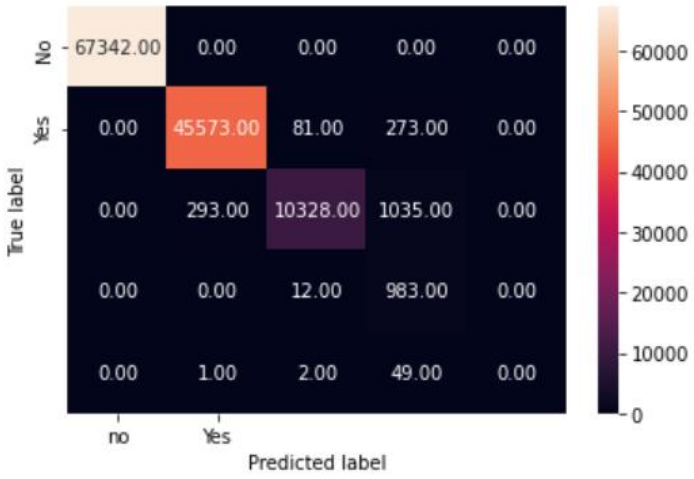

Figure 7.3 : K-nearest neighbor confusion matrix

Ada boost :

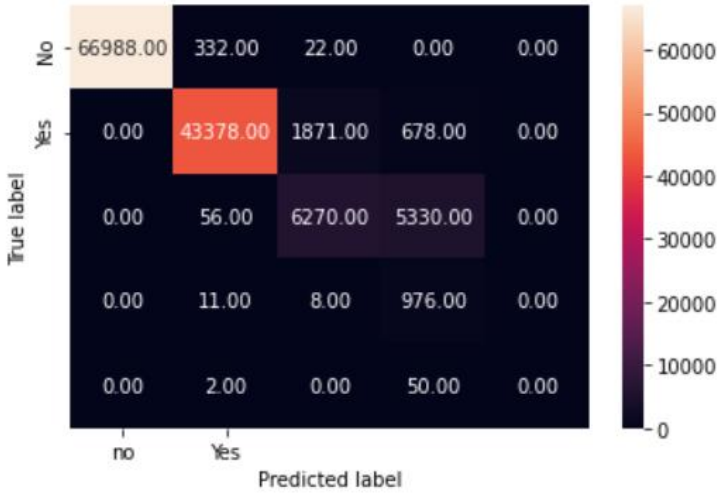

Figure 7.4 : Ada boost confusion matrix

Random forest:

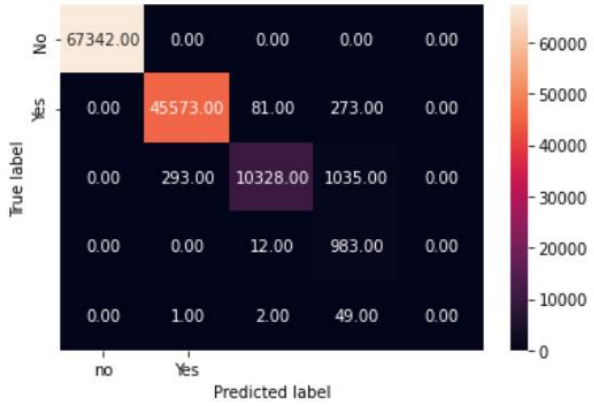

Figure 7.5 : Random forest confusion matrix

Support verctor machine:

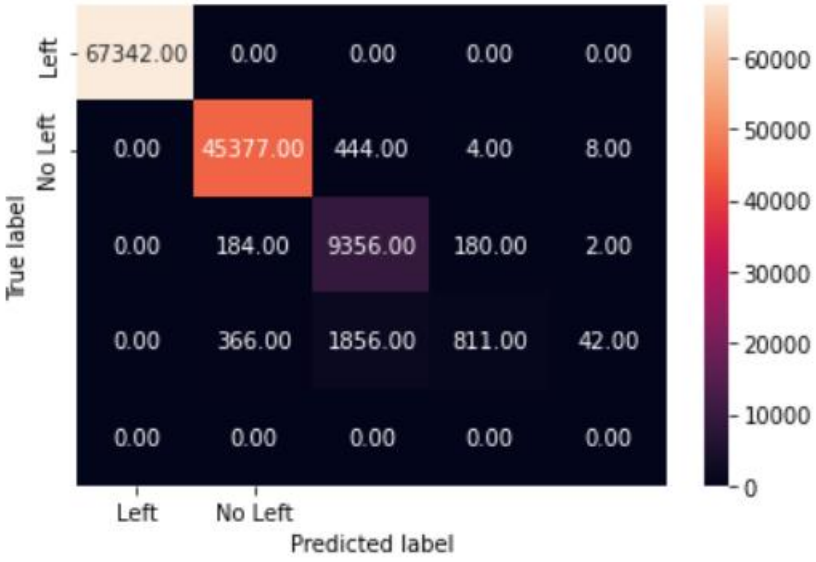

Figure 7.6 : Support verctor machine confusion matrix

Decision Tree:

Figure 7.6 : Decision Tree confusion matrix

The confusion matrix shows that the classifiers have correctly predicted a good number of attacks reviews with minor errors in dataset. Based on these figures, we can see that we have created acceptable machine learning models.

7.3 Applying Deep Learning

We applied a deep learning algorithms for our datasets: Deep neural network Architecture,

In this paper, the learning is kept constant at 0.01 while the other parameters where optimized. The count of the neurons in a layer was experimented by changing it over the range of 2 to 1024. After that, the count was further increased to 1280 but didn't yield any appreciable increase in accuracy. Therefore the neuron count was tuned to 1024.

Conventionally, increasing the count of the layers results in better results compared to increasing the neuron count in a layer. Therefore, the following network topologies were used in order to scrutinize and conclude the optimum network structure for our input data.

DNN with 1,2,3,4,5 layers. For all the above network topologies, 100 epochs were run and the results were observed. Finally, the best performance was showed by DNN 3 layer compared to all the others. To broaden the search for better results, all the common classical machine learning algorithms were used and the results were compared to the DNN 3 layer, which still

outperformed every single classical algorithm. The detailed statistical results for different network structures are reported in the table 3

Table-3

NETWORK STRUCTURE INFORMATION

Layer (type)	Output Shape	Param
Dense-1 (Dense)	(NIL, 1024)	43008
Dropout-1 (Dropout)	(NIL, 1024)	0
Dense-2 (Dense)	(NIL, 768)	787200
Dropout-2 (Dropout)	(NIL, 768)	0
Dense-3 (Dense)	(NIL, 512)	393728
Dropout-3 (Dropout)	(NIL, 512)	0
Dense-4 (Dense)	(NIL, 256)	131328
Dropout-4 (Dropout)	(NIL, 256)	0
Dense-5 (Dense)	(NIL, 128)	32896
Dropout-5 (Dropout)	(NIL, 128)	0
Dense-6 (Dense)	(NIL, 1)	129
Activation-1 (Activation)	(NIL, 1)	0

7.3.1 Proposed Architecture

An overview of proposed DNNs architecture for all use cases is shown in Fig. 1. This comprises of a hidden-layer count of 5 and an output-layer. The input-layer consists of 41 neurons. The neurons in input-layer to hidden-layer and hidden to output-layer are connected completely. Back-propagation mechanism is used to train the DNN networks. The proposed network is composed of fully connected layers, bias layers and dropout layers to make the network more robust.

Input and hidden layers: This layer consists of 41 neurons. These are then fed into the hidden layers. Hidden layers use ReLU as the non-linear activation function. Then weights are added to feed them forward to the next hidden layer. The neuron count in each hidden layer is decreased steadily from the first to the output to make the outputs more accurate and at the same time reducing the computational cost. Regularization: To make the whole process efficient and time-saving, Dropout (0.01). The function of the dropout is to unplug the neurons randomly, making the model more robust and hence preventing it from over-fitting the training set.

Output layer and classification: The out layer consists only of two neurons Attack and Benign. Since the 1024 neurons from the previous layer must be converted into just 2 neurons, a sigmoid activation function is used. Due to the nature of the sigmoid function, it returns only two outputs, hence favouring the binary classification that was intended in this paper.

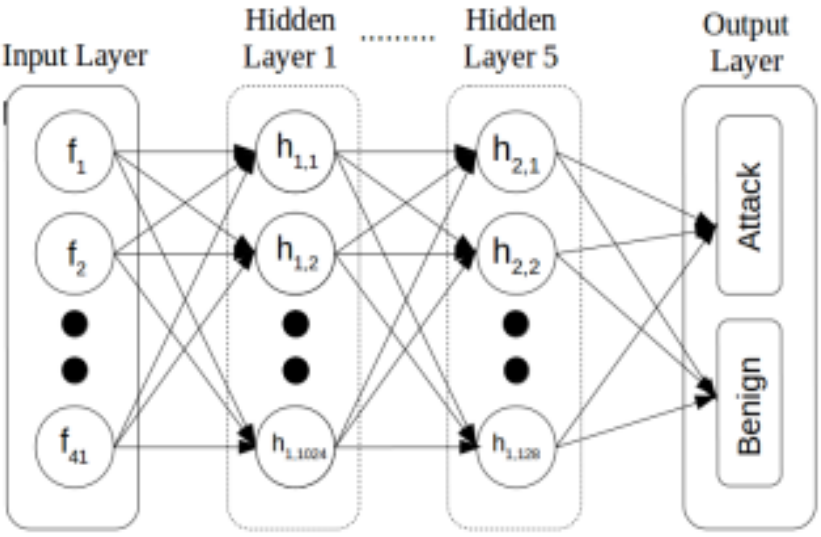

Figure 7.7 : Proposed Architecture

DNN-1:

```
Epoch 94: loss did not improve from 0.00044
157/157 [==============================] - 1s 5ms/step - loss: 4.7374e-04 - accuracy: 0.9999
Epoch 95/100
149/157 [===========================>..] - ETA: 0s - loss: 4.9107e-04 - accuracy: 0.9999
Epoch 95: loss did not improve from 0.00044
157/157 [==============================] - 1s 5ms/step - loss: 4.6928e-04 - accuracy: 0.9999
Epoch 96/100
152/157 [============================>.] - ETA: 0s - loss: 4.6411e-04 - accuracy: 0.9999
Epoch 96: loss did not improve from 0.00044
157/157 [==============================] - 1s 5ms/step - loss: 4.5196e-04 - accuracy: 0.9999
Epoch 97/100
154/157 [============================>.] - ETA: 0s - loss: 4.4639e-04 - accuracy: 0.9999
Epoch 97: loss did not improve from 0.00044
157/157 [==============================] - 1s 6ms/step - loss: 4.4116e-04 - accuracy: 0.9999
Epoch 98/100
151/157 [===========================>..] - ETA: 0s - loss: 4.2814e-04 - accuracy: 0.9999
Epoch 98: loss improved from 0.00044 to 0.00042, saving model to kddresults/dnn1layer/checkpoint-98.hdf5
157/157 [==============================] - 1s 5ms/step - loss: 4.1636e-04 - accuracy: 0.9999
Epoch 99/100
152/157 [============================>.] - ETA: 0s - loss: 4.2408e-04 - accuracy: 0.9999
Epoch 99: loss improved from 0.00042 to 0.00042, saving model to kddresults/dnn1layer/checkpoint-99.hdf5
157/157 [==============================] - 1s 5ms/step - loss: 4.1516e-04 - accuracy: 0.9999
Epoch 100/100
152/157 [============================>.] - ETA: 0s - loss: 4.1030e-04 - accuracy: 0.9999
Epoch 100: loss improved from 0.00042 to 0.00040, saving model to kddresults/dnn1layer/checkpoint-100.hdf5
157/157 [==============================] - 1s 5ms/step - loss: 3.9950e-04 - accuracy: 0.9999
```

Figure 7.8: DNN-1

```
[36] model.summary()

Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_11 (Dense)            (None, 1024)              43008

 dropout_7 (Dropout)         (None, 1024)              0

 dense_12 (Dense)            (None, 1)                 1025

 activation_4 (Activation)   (None, 1)                 0

=================================================================
Total params: 44,033
Trainable params: 44,033
Non-trainable params: 0
```

Figure 7.9:Model summary

DNN-2:

```
Epoch 93/100
155/157 [============================>.] - ETA: 0s - loss: 1.8053e-04 - accuracy: 0.9999
Epoch 93: loss improved from 0.00019 to 0.00018, saving model to kddresults/dnn2layer/checkpoint-93.hdf5
157/157 [==============================] - 3s 20ms/step - loss: 1.7909e-04 - accuracy: 0.9999
Epoch 94/100
155/157 [============================>.] - ETA: 0s - loss: 1.1720e-04 - accuracy: 0.9999
Epoch 94: loss improved from 0.00018 to 0.00012, saving model to kddresults/dnn2layer/checkpoint-94.hdf5
157/157 [==============================] - 3s 20ms/step - loss: 1.1641e-04 - accuracy: 0.9999
Epoch 95/100
156/157 [============================>.] - ETA: 0s - loss: 7.9199e-05 - accuracy: 1.0000
Epoch 95: loss improved from 0.00012 to 0.00008, saving model to kddresults/dnn2layer/checkpoint-95.hdf5
157/157 [==============================] - 3s 20ms/step - loss: 7.9073e-05 - accuracy: 1.0000
Epoch 96/100
156/157 [============================>.] - ETA: 0s - loss: 1.9590e-05 - accuracy: 1.0000
Epoch 96: loss improved from 0.00008 to 0.00002, saving model to kddresults/dnn2layer/checkpoint-96.hdf5
157/157 [==============================] - 3s 20ms/step - loss: 1.9558e-05 - accuracy: 1.0000
Epoch 97/100
155/157 [============================>.] - ETA: 0s - loss: 2.3674e-05 - accuracy: 1.0000
Epoch 97: loss did not improve from 0.00002
157/157 [==============================] - 3s 20ms/step - loss: 2.3485e-05 - accuracy: 1.0000
Epoch 98/100
156/157 [============================>.] - ETA: 0s - loss: 1.1132e-05 - accuracy: 1.0000
Epoch 98: loss improved from 0.00002 to 0.00001, saving model to kddresults/dnn2layer/checkpoint-98.hdf5
157/157 [==============================] - 3s 20ms/step - loss: 1.1114e-05 - accuracy: 1.0000
Epoch 99/100
```

Figure 7.10 : DNN-2

```
[43] model.summary()

Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_13 (Dense)            (None, 1024)              43008

 dropout_8 (Dropout)         (None, 1024)              0

 dense_14 (Dense)            (None, 768)               787200

 dropout_9 (Dropout)         (None, 768)               0

 dense_15 (Dense)            (None, 1)                 769

 activation_5 (Activation)   (None, 1)                 0

=================================================================
Total params: 830,977
Trainable params: 830,977
Non-trainable params: 0
_________________________________________________________________
```

Figure 7.10: DNN-2 model summary

DNN-3:

```
Epoch 89/100
156/157 [============================>.] - ETA: 0s - loss: 5.2307e-04 - accuracy: 0.9999
Epoch 89: loss did not improve from 0.00025
157/157 [==============================] - 4s 28ms/step - loss: 5.2224e-04 - accuracy: 0.9999
Epoch 90/100
156/157 [============================>.] - ETA: 0s - loss: 3.8292e-04 - accuracy: 0.9999
Epoch 90: loss did not improve from 0.00025
157/157 [==============================] - 4s 28ms/step - loss: 3.8231e-04 - accuracy: 0.9999
Epoch 91/100
156/157 [============================>.] - ETA: 0s - loss: 3.5247e-04 - accuracy: 0.9999
Epoch 91: loss did not improve from 0.00025
157/157 [==============================] - 4s 28ms/step - loss: 3.5191e-04 - accuracy: 0.9999
Epoch 92/100
156/157 [============================>.] - ETA: 0s - loss: 2.9923e-04 - accuracy: 0.9999
Epoch 92: loss did not improve from 0.00025
157/157 [==============================] - 4s 28ms/step - loss: 2.9875e-04 - accuracy: 0.9999
Epoch 93/100
156/157 [============================>.] - ETA: 0s - loss: 3.3332e-04 - accuracy: 0.9999
Epoch 93: loss did not improve from 0.00025
157/157 [==============================] - 4s 28ms/step - loss: 3.3279e-04 - accuracy: 0.9999
Epoch 94/100
155/157 [============================>.] - ETA: 0s - loss: 3.2856e-04 - accuracy: 0.9999
Epoch 94: loss did not improve from 0.00025
157/157 [==============================] - 4s 28ms/step - loss: 3.2593e-04 - accuracy: 0.9999
Epoch 95/100
```

Figure 7.11: DNN-3

```
[46] model.summary()

Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_16 (Dense)            (None, 1024)              43008

 dropout_10 (Dropout)        (None, 1024)              0

 dense_17 (Dense)            (None, 768)               787200

 dropout_11 (Dropout)        (None, 768)               0

 dense_18 (Dense)            (None, 512)               393728

 dropout_12 (Dropout)        (None, 512)               0

 dense_19 (Dense)            (None, 1)                 513

 activation_6 (Activation)   (None, 1)                 0

=================================================================
Total params: 1,224,449
Trainable params: 1,224,449
Non-trainable params: 0
_________________________________________________________________
```

Figure 7.12 : DNN-3 model summary

DNN-4:

```
156/157 [============================>.] - ETA: 0s - loss: 3.8542e-04 - accuracy: 0.9999
Epoch 89: loss did not improve from 0.00023
157/157 [==============================] - 5s 31ms/step - loss: 3.8481e-04 - accuracy: 0.9999
Epoch 90/100
157/157 [==============================] - ETA: 0s - loss: 3.8365e-04 - accuracy: 0.9999
Epoch 90: loss did not improve from 0.00023
157/157 [==============================] - 5s 31ms/step - loss: 3.8365e-04 - accuracy: 0.9999
Epoch 91/100
156/157 [============================>.] - ETA: 0s - loss: 3.9306e-04 - accuracy: 0.9999
Epoch 91: loss did not improve from 0.00023
157/157 [==============================] - 5s 32ms/step - loss: 3.9243e-04 - accuracy: 0.9999
Epoch 92/100
156/157 [============================>.] - ETA: 0s - loss: 2.4095e-04 - accuracy: 0.9999
Epoch 92: loss did not improve from 0.00023
157/157 [==============================] - 5s 32ms/step - loss: 2.4059e-04 - accuracy: 0.9999
Epoch 93/100
156/157 [============================>.] - ETA: 0s - loss: 5.6690e-04 - accuracy: 0.9999
Epoch 93: loss did not improve from 0.00023
157/157 [==============================] - 5s 31ms/step - loss: 5.6636e-04 - accuracy: 0.9999
Epoch 94/100
156/157 [============================>.] - ETA: 0s - loss: 2.5477e-04 - accuracy: 0.9999
Epoch 94: loss did not improve from 0.00023
157/157 [==============================] - 5s 31ms/step - loss: 2.5438e-04 - accuracy: 0.9999
Epoch 95/100
156/157 [============================>.] - ETA: 0s - loss: 4.2753e-04 - accuracy: 0.9999
Epoch 95: loss did not improve from 0.00023
```

Figure 7.13: DNN-4

```
[49] model.summary()

Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_20 (Dense)            (None, 1024)              43008

 dropout_13 (Dropout)        (None, 1024)              0

 dense_21 (Dense)            (None, 768)               787200

 dropout_14 (Dropout)        (None, 768)               0

 dense_22 (Dense)            (None, 512)               393728

 dropout_15 (Dropout)        (None, 512)               0

 dense_23 (Dense)            (None, 256)               131328

 dropout_16 (Dropout)        (None, 256)               0

 dense_24 (Dense)            (None, 1)                 257

 activation_7 (Activation)   (None, 1)                 0

=================================================================
Total params: 1,355,521
Trainable params: 1,355,521
Non-trainable params: 0
_________________________________________________________________
```

Figure 7.14 : DNN-4 model summary

DNN-5:

```
Epoch 88/100
157/157 [==============================] - ETA: 0s - loss: 4.2048e-04 - accuracy: 0.9999
Epoch 88: loss did not improve from 0.00032
157/157 [==============================] - 5s 32ms/step - loss: 4.2048e-04 - accuracy: 0.9999
Epoch 89/100
155/157 [============================>.] - ETA: 0s - loss: 2.8453e-04 - accuracy: 0.9999
Epoch 89: loss improved from 0.00032 to 0.00028, saving model to kddresults/dnn5layer/checkpoint-89.hdf5
157/157 [==============================] - 5s 32ms/step - loss: 2.8226e-04 - accuracy: 0.9999
Epoch 90/100
157/157 [==============================] - ETA: 0s - loss: 5.7507e-04 - accuracy: 0.9999
Epoch 90: loss did not improve from 0.00028
157/157 [==============================] - 5s 32ms/step - loss: 5.7507e-04 - accuracy: 0.9999
Epoch 91/100
155/157 [============================>.] - ETA: 0s - loss: 3.4498e-04 - accuracy: 0.9999
Epoch 91: loss did not improve from 0.00028
157/157 [==============================] - 5s 32ms/step - loss: 3.4289e-04 - accuracy: 0.9999
Epoch 92/100
156/157 [============================>.] - ETA: 0s - loss: 3.0227e-04 - accuracy: 0.9999
Epoch 92: loss did not improve from 0.00028
157/157 [==============================] - 5s 31ms/step - loss: 3.0178e-04 - accuracy: 0.9999
Epoch 93/100
155/157 [============================>.] - ETA: 0s - loss: 6.5598e-04 - accuracy: 0.9998
Epoch 93: loss did not improve from 0.00028
157/157 [==============================] - 5s 31ms/step - loss: 6.5074e-04 - accuracy: 0.9998
Epoch 94/100
155/157 [============================>.] - ETA: 0s - loss: 0.0019 - accuracy: 0.9996
Epoch 94: loss did not improve from 0.00028
157/157 [==============================] - 5s 31ms/step - loss: 0.0019 - accuracy: 0.9996
Epoch 95/100
```

Figure 7.15: dnn-5

```
[52] model.summary()

Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_25 (Dense)            (None, 1024)              43008

 dropout_17 (Dropout)        (None, 1024)              0

 dense_26 (Dense)            (None, 768)               787200

 dropout_18 (Dropout)        (None, 768)               0

 dense_27 (Dense)            (None, 512)               393728

 dropout_19 (Dropout)        (None, 512)               0

 dense_28 (Dense)            (None, 256)               131328

 dropout_20 (Dropout)        (None, 256)               0

 dense_29 (Dense)            (None, 128)               32896

 dropout_21 (Dropout)        (None, 128)               0

 dense_30 (Dense)            (None, 1)                 129

 activation_8 (Activation)   (None, 1)                 0

=================================================================
Total params: 1,388,289
Trainable params: 1,388,289
Non-trainable params: 0
_________________________________________________________________
```

Figure 7.16: DNN-5 model summary

7.4 DNN layering result:

Table-3

Algorithm	Accuracy	precision	Recall	F1 score
DNN-1	0.929	0.998	0.915	0.954
DNN-2	0.929	0.998	0.914	0.954
DNN-3	0.930	0.997	0.915	0.955
DNN-4	0.929	0.999	0.913	0.954
DNN-5	0.927	0.998	0.911	0.953

7.5 Result analysis:

For the scope of this paper, the KDDCup-'99' dataset was fed into classical ML algorithms as well as DNNs of varying hidden layers. After the training is completed, all models were compared for f1-score, accuracy, recall and precision with the test dataset. The scores for the same has been compared in detail in Table II. DNN 3 layer network has outperformed all the other classical machine learning algorithms. It is so because of the ability of DNNs to extract data and features with higher abstraction and the non-linearity of the networks adds up to the advantage when compared with the other algorithms.

7.5.1 Final Comparison of ML and DL

For our thesis paper, we have applied both machine learning and deep learning algorithms for network intrusion detection. With machine learning algorithms, we have achieved quite good test accuracies particularly with Random forest and Logistic Regression with accuracies of 92.829% and 92.318% for our Dataset respectively with higher precision, recall and f-1 score. But the implementation of deep learning on these datasets gives us better test accuracy then machine learning particularly Dense Layer -3 (DNN-3) Architecture with Validation Accuracy of 93.00%. So, if compare these results based on our finding on these particular datasets, we can come up to this conclusion that Dense Layer Architecture as well as deep learning algorithms represents the suitable method for network intrusion detection system.

Chapter EIGHT

8.1 Research challenges

This subsection highlights the research challenges in the field of IDS.

The current study brought to light the absence of a current dataset that reflects newer attacks for modern networks. Because these models were not sufficiently trained with enough attack kinds and patterns, the majority of the offered approaches were unable to detect zero-day attacks. An effective IDS model must be developed, tested, and confirmed using a dataset that includes both old and new threats. The ML/DL model will be able to learn more patterns and, eventually, will be able to protect against the maximum number of intrusion of various sorts by including the maximum number of attacks description in a dataset.Another research challenge for IDS is their performance in the real-world environment. Since most of the proposed methodologies are tested and verified within a lab using the public datasets. None of the proposed methodologies is tested in a real-world environment. So, it is still not clear how they will perform in real-world scenarios. As stated, most of them still rely on testing using old datasets. So the biggest challenge for the proposed methodology is to be as efficient as demonstrated in the lab tests. The proposed method once tested in the lab should also be tested in a real-time environment to verify its effectiveness for modern networks.

8.2 Future works

For improving the performance of the techniques which we have used, we will continue our research work in future, and we planned to propose some algorithms for detecting the Network intrusion . We are also interested in applying RNN and LSTM on our datasets to see how it performs. Moreover, we want to extend this work by performing similar analysis on a completely different dataset such as NF-UNSW-NB15 and CICIDS-17. By applying newer technique for we hope to get one step closer towards building an better network intrusion detection system. We also hope this study provides a baseline for the future tests and broadens scope of the solutions dealing with a sustainable NIDS .

8.3 CONCLUSIONS

This paper provides an extensive review of the network intrusion detection mechanisms based on the ML and DL methods to provide the new researchers with the updated knowledge, recent trends, and progress of the field. A systematic approach is adopted for the selection of the relevant articles in the field of AI-based NIDS. Firstly, the concept of IDS and its different classification schemes is elaborated extensively based on the reviewed articles. Then the methodology of each article is discussed and the strengths and weaknesses of each are highlighted in terms of the intrusion detection capability and complexity of the model. Based on this study, the recent trend reveals the usage of DL-based methodologies to improve the performance and effectiveness of NIDS in terms of detection accuracy and reduction in FAR. In

ML-based method using Logistic Regression we got 92.31%, GNB 92.98%, RF 92.82%, Decision Tree 91.61% which is quite good accuracy. But in case of Deep learning we applied DNN of 5 layers there we got a better accuracy which is 93.00% into DNN 3rd layer. So both ML and DL performed well in case of NIDS. For future research, we will use this knowledge to design a novel, lightweight, and efficient DL-based NIDS which will effectively detect the intruders within the network.

Reference

[1] https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/

[2] Abdulbasit Ahmed, Alexei Lisitsa, and Clare Dixon. A misuse-based network intrusion detection system using temporal logic and stream processing. *2011 5th International Conference on Network and System Security*, 2011.

[3] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer. Stateful intrusion detection for high-speed networks. *Proceedings 2002 IEEE Symposium on Security and Privacy*.

[4] Tran and Huy Nhut. A dynamic scalable parallel network-based intrusion detection system using intelligent rule ordering, Aug 2017.

[5] Christopher V. Kopek, Errin W. Fulp, and Patrick S. Wheeler. Distributed data parallel techniques for content-matching intrusion detection systems. *MILCOM 2007 - IEEE Military Communications Conference*, 2007.

[6] Sbrownlee, "supervised and unsupervised machine learning algorithms," machine learning mastery, 15-mar-2016., Sep 2016.

[7] Tadashi Ogino. Evaluation of machine learning method for intrusion detection system. International Journal of Machine Learning and Computing,
5(2):137–141, 2015.

[8] Symantec: Spammers using adwords. Network Security, 2008(5):20, 2008.

[9] https://www.gi-de.com/en/spotlight/digital-infrastructures/cybersecurity-vulnerabilities?gclid=EAIaIQobChMI5Mjy0oSa_AIVl5lmAh1uNQwpEAAYASAAEgLAhfDBwE#:~:text=Between%20January%20and,digital%20infrastructures%20now.%E2%80%9D

[10] (National Statistics 2008). " National Statistics. Internet Access, 2008. Available online: http://www.statistics.gov.uk/cci/nugget.asp?id=8 [Accessed May 8th 2009] .

[11] ’’A.Marx. 2008. Malware vs. Anti-Malware: (How) Can We Still Survive? Virus Bulletin, 2, 2.

[12] https://www.comparitech.com/antivirus/malware-statistics-facts/

[13] “D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver. 2003. Inside the Slammer Worm. IEEE Security and Privacy, 1, 33–39.

[14] “BBC. Cyber-security strategy launched, 2009a. Available online: http://news.bbc.co.uk/1/hi/uk_ politics/8118348.stm [Accessed June 25th 2009].
[15] BBC. Major cyber spy network uncovered, 2009b. Available online: http://news.bbc.co.uk/1/hi/ world/americas/7970471.stm [Accessed May 8th 2009].

[16] BBC. Spies ’infiltrate US power grid’, 2009c. Available online: http://news.bbc.co.uk/1/hi/ technology/7990997.stm [Accessed May 8th 2009].

[17] BBC. US launches cyber security plan , 2009d. Available online: http://news.bbc.co.uk/1/hi/world/ americas/8073654.stm [Accessed June 8th 2009].

[18] “C. Kruegel, F. Valeur and G. Vigna. Intrusion Detection and Correlation: Challenges and Solutions. Springer-Verlag Telos, 2004.

[19] “D. Gollmann. Computer Security. Wiley, 2nd edition, 2006

[20] “C. Kruegel, F. Valeur and G. Vigna. Intrusion Detection and Correlation: Challenges and Solutions. Springer-Verlag Telos, 2004.

[21] “R.A. Kemmerer and G. Vigna. April 2002. Intrusion Detection: A Brief History and Overview. Computer, 35, 27–30

[22] (Lewis 1993, Owens and Levary 2006).“S.F. Owens and R.R. Levary. 2006. An adaptive expert system approach for intrusion detection. Int. J. Secur. Netw., 1, 206–217. ISSN 1747-8405.

[23] C.Kruegel, F. Valeur and G. Vigna. Intrusion Detection and Correlation: Challenges and Solutions. Springer-Verlag Telos, 2004.

[24] https://www.grin.com/document/469095“C. Kruegel, F. Valeur and G. Vigna. Intrusion Detection and Correlation: Challenges and Solutions. Springer-Verlag Telos, 2004.

[25] “The UCI KDD Archive. Kdd cup 1999 data, 1999. Available online: http://kdd.ics.uci.edu/ databases/kddcup99/kddcup99.html [Accessed March 5th 2010]

[26]Y . Bouzida and F. Cuppens. Detecting Known and Novel Network Intrusions. In IFIP/SEC 2006, 21st IFIP TC-11 International Information Security Conference, 2006a.

[27] R. Lippmann, J. Haines, D. Fried, J. Korba and K. Das. ”The 1999 DARPA off-line intrusion detection evaluation”. Computer networks, vol. 34, no. 4, pp. 579 595, 2000. DOI http://dx.doi.org/10.1016/S1389- 1286(00)00139-0.

[28] W. Lee and S. Stolfo. ”A framework for constructing features and models for intrusion detectionsystems”. ACM transactions on information and system security, vol. 3, no. 4, pp. 227261, 2000. DOI http://dx.doi. Org/10.1145/382912.382914.

[29] B. Pfahringer. ”Winning the KDD99 classification cup: Bagged boosting”. SIGKDD explorations newsletter, vol. 1, pp. 6566, 2000. DOI http://dx.doi.org/10 . 1145/846183.846200 .

[30] M. Vladimir, V. Alexei and S. Ivan. ”The MP13 approach to the KDD’99 classifier learning contest”. SIGKDD explorations newsletter, vol. 1, pp. 76 77, 2000. DOI http://dx.doi.org/ 10.1145/846183. 846202.

[31] R. Agarwal and M. Joshi. ”PNrule: A new framework for learning classier models in data mining”. Tech. Rep. 00-015, Department of Computer Science, University of Minnesota, 2000.

[32] C. Elkan. ”Results of the KDD’99 classifier learning”. SIGKDD explorations newsletter, vol. 1, pp. 63 64, 2000. DOI http://dx.doi.org/10.1145/846183. 846199 .

[33] S. Sung, A.H. Mukkamala. ”Identifying important features for intrusion detection using support vector machines and neural networks”. In Proceedings of the symposium on applications and the Internet (SAINT), pp. 209216. IEEE Computer Society, 2003. DOI http: //dx.doi.org/10.1109/saint. 2003.1183050.

[34] H. Kayacik, A. Zincir-Heywood and M. Heywood. ”Selecting features for intrusion detection: A feature relevance analysison KDD 99 intrusion detection datasets”. In Proceedings of the third annual conference on privacy, security and trust (PST). 2005.

[35] C. Lee, S. Shin and J. Chung. ”Network intrusion detection through genetic feature selection”. In Seventh ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing (SNPD), pp. 109114. IEEE Computer Society, 2006

[36] S. Chavan, K. Shah, N. Dave, S. Mukherjee, A. Abraham and S. Sanyal. ”Adaptive neuro-fuzzy intrusion detection systems”. In Proceedings of the international conference on information technology: Coding and computing (ITCC), vol. 1, pp. 7074. IEEE Computer Society, 2004. DOI http://dx.doi.org/ 10.1109/ itcc. 2004. 1286428.

[37] S. Chebrolu, A. Abraham and J. Thomas. ”Feature deduction and ensemble design of intrusion detection systems”. Computers & security, vol. 24, no. 4, pp. 295307, 2005. DOI http://dx.doi.org/10.1016/j. Cose.2004.09.008.

[38] Y. Chen, A. Abraham and J. Yang. ”Feature selection and intrusion detection using hybrid flexible neural tree”. In Advances in neural networks (ISNN), vol. 3498 of Lecture notes in computer science, pp. 439 444. Springer Berlin / Heidelberg, 2005. DOI http: //dx.doi.org/10.1007/11427469 71.

[39] C. Sinclair, L. Pierce and S. Matzner. ”An application of machine learning to network intrusion detection”. In Proceedings of the 15th annual computer security applications conference (ACSAC), pp. 371377. IEEE Computer Society, 1999. DOI http://dx.doi.org/ 10.1109/csac.1999.816048.

[40] H. Debar, M. Becker and D. Siboni. ”A neural network component for an intrusion detection system”. In Proceedings of the IEEE computer society symposium on research in security and privacy, pp. 240250. IEEE Computer Society, 1992. DOI http://dx.doi.org/ 10.1109/risp.1992.213257.

[41] J. Cannady. ”Artificial neural networks for misuse detection”. In Proceedings of the 1998 national information systems security conference (NISSC), pp. 443456. Citeseer, 1998.

[42] H. Debar and B. Dorizzi. ”An application of a recurrent network to an intrusion detection system”. In International joint conference on neural networks, 1992. IJCNN., vol. 2, pp. 478 483 vol.2. jun 1992. DOI

http://dx.doi.org/10.1109/ijcnn. 1992.226942.

[43] Z. Zhang, J. Lee, C. Manikopoulos, J. Jorgenson and J. Ucles. ”Neural networks in statistical anomaly intrusion detection”. Neural network world, vol. 11, no. 3, pp. 305316, 2001.

[44] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani. ”A detailed analysis of the KDD CUP 99 data set”. In IEEE symposium on computational intelligence for security and defense applications, Cisda, pp. 16. IEEE, Jul. 2009. DOI http://dx.doi.org/10.1109/cisda. 2009.5356528

[45] J. McHugh. ”Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory”. ACM transactions on information and system security, vol. 3, no. 4, pp. 262294, 2000. DOI http://dx.doi.org/10.1145/382912.382923

[46] R. Agarwal and M. V. Joshi, Pnrule: A new framework for learning classifier models in data mining (a case-study in network intrusion detection), in Proceedings of the 2001 SIAM International Conference on Data Mining. SIAM, 2001, pp. 117.

[47] Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017, September). Applying convolutional neural network for network intrusiondetection. In Advances in Computing, Communications and Informatics(ICACCI), 2017 International Conference on (pp. 1222-1228). IEEE.

[48] Staudemeyer, R. C. (2015). Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal, 56(1), 136-154

[49] S-Y. Wu and E. Yen. 2009. Data mining-based intrusion detectors. Expert Syst. Appl., 36, 5605–5612. ISSN 0957-4174.

[50]. Kabiri P, Ghorbani AA. Research on intrusion detection and response: a survey. Int J Netw Secur. 2005;1(2):84-102. https://doi.org/10. 6633/IJNS.200509.1(2).05.

[51]. Zhang Y, Lee W, Huang YA. Intrusion detection techniques for mobile wireless networks. Wirel Netw. 2003;9(5):545-556. https://doi.org/ 10.1023/A:1024600519144.

[52]. Kabiri P, Ghorbani AA. Research on intrusion detection and response: a survey. Int J Netw Secur. 2005;1(2):84-102. https://doi.org/10. 6633/IJNS.200509.1(2).05.

[53]. Zhang Y, Lee W, Huang YA. Intrusion detection techniques for mobile wireless networks. Wirel Netw. 2003;9(5):545-556. https://doi.org/ 10.1023/A:1024600519144.

[54]. Kabiri P, Ghorbani AA. Research on intrusion detection and response: a survey. Int J Netw Secur. 2005;1(2):84-102. https://doi.org/10. 6633/IJNS.200509.1(2).05.

[55]. Uddin M, Rahman AA, Uddin N, Memon J, Alsaqour RA, Kazi S. Signature-based multi-layer distributed intrusion detection system using mobile agents. Int J Netw Secur. 2013;15(2):97-105. https://doi.org/10.6633/IJNS.201303.15(2).03.

[56]. Neri F. Comparing local search with respect to genetic evolution to detect intrusions in computer networks. Paper presented at: Proceedings of the Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512). La Jolla, CA, USA: IEEE; vol. 1, 2000:238-243.

[57]. Ma W. Analysis of anomaly detection method for Internet of things based on deep learning. Trans Emerg Telecommun Technol. 2020;e3893. https://doi.org/10.1002/ett.3893.

[58]. Zhang Z, Shen H, Sang Y. An observation-centric analysis on the modeling of anomaly-based intrusion detection. Int J Netw Secur. 2007;4(3):292-305. https://doi.org/10.6633/IJNS. 200705.4(3).08.

[59] Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. ACM Comput Surv. 2009;41(3):1-58. https://doi.org/10.1145/1541880. 1541882

[60] Mehmood Y, Ahmad F, Yaqoob I, Adnane A, Imran M, Guizani S. Internet-of-Things-based smart cities: recent advances and challenges. IEEE Commun Mag. 2017;55(9):16-24. https://doi.org/10.1109/

[61] Shah SA, Seker DZ, Hameed S, Draheim D. The rising role of big data analytics and IoT in disaster management: recent advances taxonomy and prospects. IEEE Access. 2019;7:54595-54614. https://doi.org/10.1109/ACCESS.2019.2913340.

[62] Lazarescu MT. Wireless sensor networks for the Internet of Things: barriers and synergies. Components and Services for IoT Platforms. New York, NY: Springer; 2017:155-186.

[63] Haseeb K, Almogren A, Islam N, Ud Din I, Jan Z. An energy-efficient and secure routing protocol for intrusion avoidance in IoT-based WSN. Energies. 2019;12(21):4174. https://doi.org/10.3390/en12214174.

[64] Krzyszton M, Marks M. Simulation of watchdog placement for cooperative anomaly detection in bluetooth mesh intrusion detection ´ system. Simul Model Pract Theory. 2020;101:102041. https://doi.org/10.1016/j.simpat.2019.102041

[65] Shen S, Yue G, Cao Q, Yu F. A survey of game theory in wireless sensor networks security. J Netw. 2011;6(3):521. https://doi.org/10.4304/ jnw.6.3.521-532.

[66] Abdalzaher MS, Muta O. A game-theoretic approach for enhancing security and data trustworthiness in IoT applications. IEEE IoT J. 2020. https://doi.org/10.1109/JIOT. 2020.2996671

[67] Khan ZA, Herrmann P. A trust based distributed intrusion detection mechanism for internet of things. Paper presented at: Proceedings of the IEEE 31st International Conference on Advanced Information Networking and Applications (AINA). Taipei, Taiwan: IEEE; 2017:1169-1176.

[68] Ahmad F, Kurugollu F, Adnane A, Hussain R, Hussain F. MARINE: man-in-the-middle attack resistant trust model in connected vehicles. IEEE IoT J. 2020;7(4):3310-3322. https://doi.org/10.1109/JIOT.2020.2967568

[69] Abdalzaher MS, Muta O. Employing game theory and TDMA protocol to enhance security and manage power consumption in wsns-based cognitive radio. IEEE Access. 2019;7:132923-132936. https://doi.org/10.1109/ACCESS.2019.2940699.

[70] Abdalzaher MS, Seddik K, Muta O. Using repeated game for maximizing high priority data trustworthiness in wireless sensor networks. Paper presented at: Proceedings of the IEEE Symposium on Computers and Communications (ISCC). Heraklion, Greece: IEEE;2017:552-557

[71] Asaka *et al.* (1999) and Kruegel *et al.* (2004, p. 4) offer the following breakdown of a successful intrusion:

[72] Arifuzzaman, M., & Matsumoto, M. (2012). An efficient medium access control protocol with parallel transmission for wireless sensor networks. *Journal of Sensor and Actuator Networks*, *1*(2), 111-122.

[73].S. Benferhat, F. Autrel and F. Cuppens. Enhanced Correlation in an Intrusion Detection Process. In Lecture Notes in Computer Science, volume 2776, pages 157–170. Springer, 2003.

[74] Arifuzzaman, M., Yu, K., & Sato, T. (2014, June). Content distribution in Information Centric Network: Economic incentive analysis in game theoretic approach. In *Proceedings of the 2014 ITU kaleidoscope academic conference: Living in a converged world-Impossible without standards?* (pp. 215-220). IEEE.

[75] Arifuzzaman, M., Keping, Y., Nguyen, Q. N., & Takuro, S. (2015, June). Locating the content in the locality: ICN caching and routing strategy revisited. In *2015 European Conference on Networks and Communications (EuCNC)* (pp. 423-428). IEEE.

[76] "Glenn, M. (2003). A summary of DoS/DDoS prevention, monitoring and mitigation techniques in a service provider environment. *SANS Institute*, *Aug*, *21*, 34.

[77] Arifuzzaman, M., Alam, M. S., & Matsumoto, M. (2011, December). A hybrid MAC with intelligent sleep scheduling for wireless sensor networks. In *Proceedings of ITU Kaleidoscope 2011: The Fully Networked Human?-Innovations for Future Networks and Services (K-2011)* (pp. 1-7). IEEE.

[78] Arifuzzaman, M., Dobre, O. A., Ahmed, M. H., & Ngatched, T. M. (2016, December). Joint routing and mac layer qos-aware protocol for wireless sensor networks. In *2016 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE.

[79] "Baqer, M. & Khan, A. (2007). Energy-efficient pattern recognition approach for wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, (pp. 509{514). IEEE.

[80] Nguyen, Q. N., Arifuzzaman, M., & Sato, T. (2015, December). Proactive-caching based information centric networking architecture for reliable green communication in intelligent

transport system. In *2015 ITU Kaleidoscope: Trust in the Information Society (K-2015)* (pp. 1-7). IEEE.

[81] S. P. Ripa, F. Islam and M. Arifuzzaman, "The Emergence Threat of Phishing Attack and The Detection Techniques Using Machine Learning Models," *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, Rajshahi, Bangladesh, 2021, pp. 1-6, doi: 10.1109/ACMI53878.2021.9528204

[82] T. Toma, S. Hassan and M. Arifuzzaman, "An Analysis of Supervised Machine Learning Algorithms for Spam Email Detection," 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), Rajshahi, Bangladesh, 2021, pp. 1-5, doi: 10.1109/ACMI53878.2021.9528108.

[83] Yu, K., Zhu, L., Wen, Z., Mohammad, A., Zhou, Z., & Sato, T. (2014, September). CCN-AMI: Performance evaluation of content-centric networking approach for advanced metering infrastructure in smart grid. In 2014 IEEE international workshop on applied measurements for power systems proceedings (AMPS) (pp. 1-6). IEEE.

[84] Lopez, J. E., Arifuzzaman, M., Zhu, L., Wen, Z., & Takuro, S. (2015, December). Seamless mobility in data aware networking. In 2015 ITU Kaleidoscope: Trust in the Information Society (K-2015) (pp. 1-7). IEEE.

[85] Aziz, A., Saha, S., & Arifuzzaman, M. (2021). Analyzing Banking Data Using Business Intelligence: A Data Mining Approach. In *Proceedings of International Joint Conference on Advances in Computational Intelligence: IJCACI 2020* (pp. 245-256). Springer Singapore.

[86] Nguyen, Q. N., López, J., Tsuda, T., Sato, T., Nguyen, K., Arifuzzaman, M., ... & Thanh, N. H. (2020, January). Adaptive caching for beneficial content distribution in information-centric networking. In 2020 International Conference on Information Networking (ICOIN) (pp. 535-540). IEEE.

[87] Arifuzzaman, M., Matsumoto, M., & Sato, T. (2013). An intelligent hybrid MAC with traffic-differentiation-based QoS for wireless sensor networks. IEEE sensors journal, 13(6), 2391-2399.

[88] Nguyen, Q. N., Arifuzzaman, M., Miyamoto, T., & Takuro, S. (2015, March). An optimal information centric networking model for the future green network. In *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems* (pp. 272-277). IEEE.

[89] Q. N. Nguyen, M. Arifuzzaman, K. Yu and T. Sato, "A Context-Aware Green Information-Centric Networking Model for Future Wireless Communications," in IEEE Access, vol. 6, pp. 22804-22816, 2018, doi: 10.1109/ACCESS.2018.2828462.

[90] Islam, M. S., Arifuzzaman, M., & Islam, M. S. (2019, December). SMOTE approach for predicting the success of bank telemarketing. In *2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)* (pp. 1-5). IEEE.

[91]Hasan, M. R., Maliha, M., & Arifuzzaman, M. (2019, July). Sentiment analysis with NLP on Twitter data. In 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2) (pp. 1-4). IEEE.

[92]Bhowmik, N. R., Arifuzzaman, M., Mondal, M. R. H., & Islam, M. S. (2021). Bangla text sentiment analysis using supervised machine learning with extended lexicon dictionary. Natural Language Processing Research, 1(3-4), 34-45.

[93] Bhowmik, N. R., Arifuzzaman, M., & Mondal, M. R. H. (2022). Sentiment analysis on Bangla text using extended lexicon dictionary and deep learning algorithms. Array, 13, 100123.

[94] Gope, J. C., Tabassum, T., Mabrur, M. M., Yu, K., & Arifuzzaman, M. (2022, February). Sentiment Analysis of Amazon Product Reviews Using Machine Learning and Deep Learning Models. In 2022 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE) (pp. 1-6). IEEE.

[95] Nguyen, Q. N., Yu, K., Sato, T., & Arifuzzaman, M. (2017, September). A game-theoretical green networking approach for information-centric networks. In 2017 IEEE Conference on Standards for Communications and Networking (CSCN) (pp. 132-137). IEEE.

[96] Afrin, S., & Arifuzzaman, M. (2020). e-Health in developing countries: Bangladeshi perspective. Int J Eng Adv Technol (IJEAT), 9, 46.

[97] Basunia, M. R., Pervin, I. A., Al Mahmud, M., Saha, S., & Arifuzzaman, M. (2020, June). On predicting and analyzing breast cancer using data mining approach. In 2020 IEEE Region 10 Symposium (TENSYMP) (pp. 1257-1260). IEEE.

[98] Siddiq, M. A. A., Arifuzzaman, M., & Islam, M. S. (2022, March). Phishing Website Detection using Deep Learning. In Proceedings of the 2nd International Conference on Computing Advancements (pp. 83-88).