An Ensemble Machine Learning Approach for Network Intrusion Detection

By

Md. Raihan-Al-Masud

MASTER OF SCIENCE

IN

INFORMATION AND COMMUNICATION TECHNOLOGY

Institute of Information and Communication Technology

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

December, 2019

The thesis titled *"An Ensemble Machine Learning Approach for Network Intrusion Detection"* submitted by Md. Raihan-Al-Masud, Roll No.: 1015312017, Session: October 2015, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science in Information and Communication Technology on 14th December 2019.

CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

[illegible]

Md. Raihan-Al-Masud

Student ID: 1015312017

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

NIDS =Network Intrusion Detection System

CIC = Canadian Institute for Cybersecurity

DR = Detection Rate

DT = Decision Tree

RF = Random Forest

LR = Logistic Regression

SVM = Support Vector Machine

SVC = Support Vector Classifier

DoS = Denial of Service

R2L = Remote to Local

U2R = User to Root

NB = Naive Bayes

CNN = Convolutional Neural Network

ANN = Artificial Neural Network

KNN = K-nearest Neighbor

DNN = Deep Neural Network

DBN = Deep Belief Network

LSTM = Long Short Term Memory

ML = Machine Learning

IOT = Internet of Things

GA = Genetic Algorithm

RBF kernel = Radial Basis Function kernel

IP = Internet Protocol

TCP = Transmission Control Protocol

Acknowledgement

All perfect praises belong to the Almighty alone.

I would like to express my sincere gratitude to my honorable thesis supervisor Dr. Hossen Asiful Mustafa, Assistant Professor, Institute of Information and Communication Technology (IICT), Bangladesh University of Engineering and Technology (BUET), for his invaluable support, strong guidance, patience, availability and trust throughout the research work. I highly appreciate his insight, encouragement and suggestions during my research work, which helped me to build confidence and courage to overcome difficulties. I am also grateful to my course teachers of IICT, BUET for their encouragement and guidelines. Furthermore, I would like to thank all the members of the board of examiners for their precious time in understanding my work and their insightful comments.

Abstract

Due to increasing amount of cyber attack, there is a growing demand for Network intrusion detection systems (NIDSs) which are necessary for defending from potential attacks. Cyber attacks can harm in different sectors including small and large business, institutions and also an individual. By stealing confidential data and harvesting login credentials, attackers can turn an organization into destitute. Those attacks can occupy the network by traffic flooding which results in denial of service to its users. Besides, when an institution is hacked, they lose their reputation and customers. Detecting and preventing cyber attacks is one of the key research areas. Existing NIDSs use traditional machine learning algorithms with low detection rate and are also not suitable for the new unknown cyber attacks. In this thesis, we propose a detection model with ensemble machine learning methods. Ensemble method is a machine learning technique that combines several base models in order to produce one optimal predictive model. Ensemble machine learning methods have the potential to detect and prevent different types of attacks compared to traditional machine learning methods. Our proposed system uses ensemble machine learning methods with Voting, Stacking, Bagging and Boosting methodology. In this research work we have designed 16 new types of ensemble machine learning classifiers: 4 Voting, 4 Stacking, 3 Boosting, 3 Bagging and 2 Hybrid classifiers. We have used the full training and testing NSL-KDD dataset to evaluate the performance of multiclass classification and we also compare the performance with deep learning as well as traditional base level machine learning techniques. NSL-KDD dataset provides data with DoS, Probe, R2L and U2R attacks. Result shows that detection rate of DoS, Probe, R2L and U2R network attacks vary with different types of classifiers. Different classifiers perform better for different types of attacks. Moreover, we also identified that the detection rate changes with the change of the number of features. To design, develop and evaluate our proposed ensemble ML classifiers, we have used Scikit learn library which is mainly based on python language. Our proposed system can detect known attacks as well as prevent unknown attacks. Experimental results show that the proposed intrusion detection classifier is superior to the performance of existing methods. Our models can improve the detection rate of the IDS which is vital for network intrusion detection systems.

CHAPTER 1 Introduction

1.1 Introduction

Computers and networks have been under threat from viruses, worms and attacks from hackers since they were first used. Securing these devices and the data passing between them is a challenging task because the number of intrusions is also increasing sharply year by year.

1.2 Background and Motivation

In 2008, the number of devices connected to the Internet exceeded the number of human beings and this increasing trend will see about 50 billion devices by 2020 (Figure 1.1) [1].

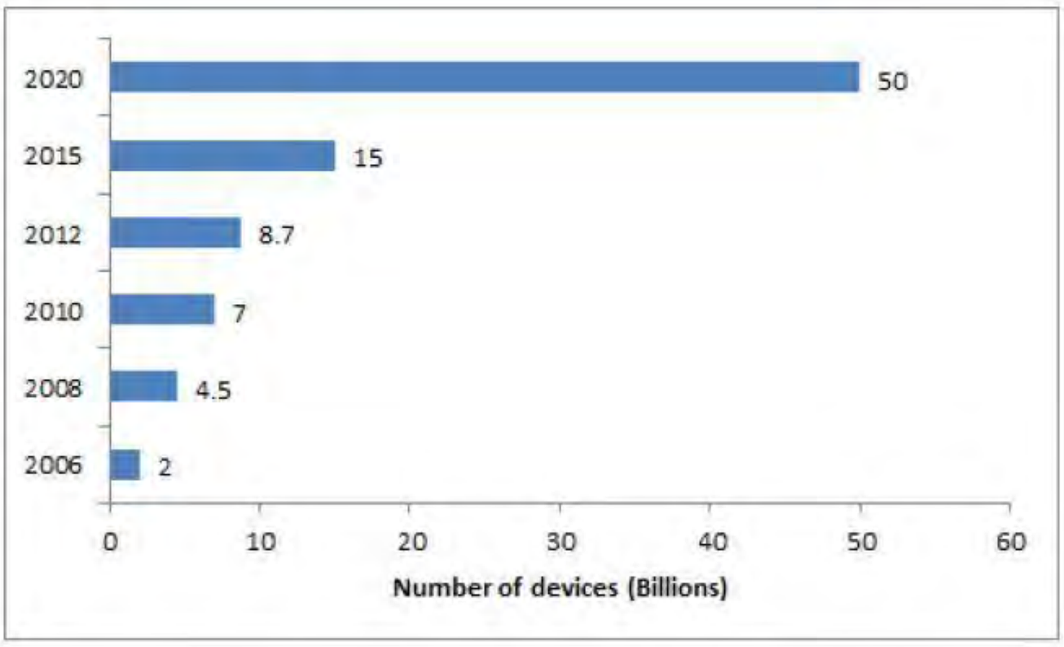


Figure 1.1: Increases on the number of devices connected to the Internet over years [1]

Furthermore, the amount of computer malware has increased rapidly in recent years from about 333,000 in 2005 to 972,000 in 2006, and 5,490,000 in 2007 [2].

When anybody is not connected to the internet, he is at some risk of intrusion due to the physical access (access by someone when the owner is not present) of any others. This threat is more when we are connected to the network, particularly the Internet. At this time, anyone can access that computer remotely to take the credential or to make an attack on that system. To exploit a system using intrusion does not need execution of manual attack by a person. It can be done by automated engineered software. A well-known example of this attack is Slammer worm. It was mainly a Denial of Service (DoS) attack in 2003 performed in Microsoft's SQL Server [3]. This attack disabled the Microsoft database and overloaded its network. This work is capable to infect approximately 75,000 computer systems within 10 minutes. Not only that, but it also can cancel airline flights, interference with elections, and ATM failures [3].

Professional companies and government organizations suffer more than a private person during cyber attacks. There are many examples of cyber attack. In 2009, using malware, the intruder was capable of shutting down an entire power grid of USA [4]. Later year, the GhostNet was discovered located in china, and it infiltrated to almost 1000 organizational computers including foreign ministries and embassies. Another government related attack was reported in 2008, when Russia launched a cyber attack against Georgia during war [5]. Due to the cyber attack to the government agencies, President Barack Obama formed a national cyber security body in the USA in May 2009 [6], followed by the UK [7].

There are several mechanisms that can be adopted to increase the security in computer systems. We can consider three levels protection:

- Attack prevention: Firewalls, usernames and passwords, and user rights.
- Attack avoidance: Encryption.
- Attack detection: Intrusion detection systems.

Despite adoption of different mechanisms, such as cryptography and protocols to make the computer network secure, it is not possible to prevent all intrusion. Firewalls can block and filter certain types of data used on a host or a network of computer. It cannot handle the misuse within a network. Thus, IDS does not replace other mechanism, but complement them when malicious behavior occurs. Before 1990, the IDS was operated by system administrator manually, but now automated IDS software system is available.

A wide range of Artificial Intelligence (AI) techniques have been adopted in IDSs, as reviewed in section 1.4. Initially, Rule Based Systems (RBSs) were the first to be employed successfully, but the drawback of RBSs is that they are inflexible (due to the rigid rules), and, thus, cannot detect new intrusions, or variations of known intrusions [8] [9].

Another area of AI, machine learning / data mining, such as base algorithms, Ensemble learning and deep learning with techniques of anomaly detection offer some desired flexibility. It can detect intrusion automatically by analyzing the behavior of user or network traffic. A benefit of this technique is that it can detect unknown or any new types of intrusion.

Recent research takes advantage of hybridization techniques to improve the detection rates of machine learning classifiers. Sabhnani et al. [10] examines 9 Machine Learning (ML) algorithms on a commonly used KDD-Cup dataset. The NSL-KDD (updated version of KDD Cup ’99) data set has been widely used to evaluate intrusion detection prototypes in the last decade.

In the literature, all studies indicate that there is a significant problem in detecting two particular classes of intrusion: *User to Root (U2R)* and *Remote to Local (R2L)*. Imbalance classes is a problem in many real life application, and has been considered in medical diagnosis [11-13], credit scoring [14], customer churn [15], [16], natural language processing [17], lexical acquisition [18] and text recognition [19]. The general problem is that the minor class (es) are not classified well when there is a significant imbalance among the classes. Artificial Neural Networks (ANNs) and Decision Trees (DTs) have been popularly applied to intrusion detection, but both have been shown to be biased towards the major class(es) [20], [21]. This corresponds with the observations in the literature on intrusion detection, in which ANNs have been reported to be unable to detect the minor class *U2R* [22] [23]. Furthermore, an alternative approach to train Ensemble Machine Learning (EML) is proposed to better learn from imbalanced data.

Most of the network-based intrusion detection research is going on base level machine learning (for example SVM, KNN, Logistic Regression, etc.) and deep learning (such as CNN, DBN, LSTM, etc.). There is very less work using Ensemble machine learning in this field. In this research our main focus in on ensemble machine learning which is one of the most powerful classifiers in the artificial intelligence domain.

This thesis presents a new set of solutions on ensemble classifier (classifier ensembles). Furthermore, the approach taken here shows a novel perspective on the analysis of the selection process for classifier ensembles and on the detection rate in the network intrusion detection systems using ensemble machine learning classifier. The Ensemble Machine Learning successfully learns from imbalanced data, which demonstrates that these classifiers are capable of detecting the minor classes with more detection rate. Another part of our research work is proposing a range of solutions with different Ensemble machine learning classifiers. Furthermore, the approach taken here shows a novel perspective on the analysis of the selection process for ensemble classifier.

1.3 Research Objective

Motivated by the present state of machine learning based network intrusion detection system, the objective of this thesis work is to develop a new design of Ensemble Machine Learning for better detection of malware in the network traffic. To fulfill this objective, the following aims have been considered:

1. To identify an order of importance of features for detecting malware by packet header inspection found within network traffic.
2. To design an efficient classifier algorithm for identifying malware in target networks based on their features.
3. To compare the algorithm with existing works in the literature.

1.4 Literature Review

Anderson, in 1980, first introduced anomaly-based intrusion detection methodology to detect abnormal activity [24]. After that, lots of researchers use machine learning to detect intrusion and make some improvements in this field. In [25], Kuang et al. proposed support vector machine (SVM) model combining with kernel trick principal with genetic algorithm (GA). K-nearest neighbor (KNN) was proposed to detect intrusion in a wireless sensor network in 2014 [26]. ANN was applied on NSL-KDD, where the detection rate was slightly higher in the classification of five classes [27]. A Random Forest (RF) model which is mainly an ensemble classifier was presented in [28]. Comparing the performance of Naive Bayes (NB) and a Decision Tree (DT), an empirical investigation was conducted on the KDD Cup '99 dataset in [29]. The DT obtains a higher accuracy (92.28% compared with 91.47%), but NB obtains better detection rates on the three minor classes, namely *Probing*, *U2R* and *R2L* intrusions. Most significantly, the DT detects merely 0.52% *R2L* intrusions whilst NB detects 7.11%. Similar observations are made by [30], as they compare NB with an ANN. ANNs and DTs are biased towards the major class(es) [20] [21], and, therefore, are prone to perform worse on the minor class(es). Therefore, this can be seen as a benefit of the NB, provided that the FPR does not become too high.

NB has also been found to be more robust than some other machine learning techniques. The performance of two probabilistic techniques, NB and a Gaussian classifier, and two predictive techniques, a DT and Random Forest (RF, an ensemble of DTs) was compared in [31]. They analyze the performance of the techniques on three different training sets of the 10% KDD Cup '99 data set (all tested on the original test set). Each training set consists of 90,000 instances, but with different proportions of normal and intrusive data. For each set, 10 randomly created versions were selected to examine the sensitivity of the techniques. In the best cases, NB and the Gaussian classifier performed significantly better on the minor classes, *U2R* and *R2L*, but NB performed worst on *DoS*. However, the DT and RF were very sensitive to the training data selected, and the mean performance was lower than the probabilistic classifiers.

The performance of NB with an Adaptive Bayesian Network (ABN) was compared in [32]. They use a different subset of the KDD Cup ’99 data set than [29], which led to significantly different results. Hence, direct comparisons are not made across these studies. However, the behavior of the algorithms is similar, *i.e.*, NB obtains higher detection rates on the minor classes. The greatest difference is clear from the detection of *U2R* and *R2L* intrusions. Due to the low proportion of instances of these two classes, the ABN does obtain the highest accuracy by correctly classifying more *Normal* and *DoS* instances, which are the major classes.

The findings of [33] suggests that NB with Kernel Estimation (NBKE) is advantageous. They compare the performance of NB with and without kernel estimation on data gathered at Wuhan University in July 2008, with a focus on detecting flooding attacks and port scans. NBKE obtains 98.80% accuracy compared with 94.40% for the basic NB algorithm. Furthermore, the authors propose using an additional feature, a Hurst exponential, which is a measure of the traffic rate and port dispersion (how many ports were used in a specific time window). Experiments on detecting UDP flooding gave a 6% higher accuracy with Hurst.

In [29] and [30], the authors motivate potential hybridizations of techniques. For example, Benferhat et al. similarly to [29], also observe that NB is better at detecting some intrusions than a DT. They emphasize on the above issue, which lead them to propose a hybrid system of anomaly detection and misuse detection.

Application of deep learning especially Deep Belies Network (DBN) was used for analysing NSL-KDD dataset [34]. In [35], Kim et al. used short term memory (LSTM) architecture to RNN, and LSTM-RNN IDS provides better accuracy with higher FAR. They used partial data from the full dataset and they also used the training set as a test set. In [36], Abolhasanzadeh et al. Proposed several methodologies using dimensionality reduction techniques on NSL-KDD. In [37], Fiore et al. presented a new technique of Discriminative Restricted Boltzmann Machine (DBM) with well classification ability. In [38], Ding et al. introduced CNN for intrusion detection and compared with different ML classifiers.

In 2018, Mirza et al. [39] implements ensemble classifier combining neural networks, decision trees and logistic regression. He used the KDD Cup 99 data set to measure the overall accuracy by combining the different types of attacks. Therefore he works with binary classification: normal traffic and attack traffic. Finally, he measures only three types of accuracy namely accuracy of attack traffic detection, accuracy of normal traffic detection and the overall accuracy.

In 2019, Hu et al. [40] proposed an ensemble technique named Dynamic Deep Forest which is a tree-based approach. In his work, there are two parts. One is Dynamic Multi-Grained Traversing where he does some feature selection work in the preprocessing stage using entropy. In this part he works with different set of features. Another part of his work is the Cascade Forest. In this part, he applied tree-based ensemble classifiers using cascades layered architecture to measure precision, recall and accuracy. In their research work, they evaluate their model using KDD'99 (KDD CUP 99) dataset.

Sharma et al. [41], in 2019, proposed an ensemble approach using ExtraTree feature selection mechanism. In their research work, at first they focused on present research scenario of intrusion detection using imbalance dataset and they mention that due to the multi-class detection using imbalance dataset, the previous work accuracy was not satisfactory. They propose three layers architecture of their work. In the first layer, they detect every attack individually and in the second layer they combine the previous layer using softmax to detect intrusion in the network. In their research work, they measure accuracy using the KDDcup99 and UNSW datasets.

In 2019, GAO et al. [111], proposed ensemble approach using NSL-KDD dataset. Among different types of classifiers, they measure detection rate when used the Random Forest ensemble classifier; and also they work with voting ensemble technique. When they work with voting ensemble classifier, they measure accuracy of their model for detecting intrusion. In their work, they set multiple decision tree and construct MultiTree classifier, and they also used different base classifiers to construct the ensemble classifiers.

In our research work, we propose ensemble machine learning approach using NSL-KDD dataset to measure the detection rate for detecting intrusion. In our work, we will compare our output with most recent research work where detection rate will be measured using ensemble machine learning approach and when they use the NSL-KDD dataset. In most recent, Hu et al. [40] and Sharma et al. [41] used the KDD CUP 99 dataset; in contrast GAO et al. [111] used RF ensemble classifier to measure the detection rate and used NSL-KDD dataset. However, all those methods have some limitations on performance, especially in detection rate.

1.5 Contribution

There are three empirical parts to this thesis. This research has made contributions to both the intrusion detection and machine learning domains. Although the focus of this thesis is on the application of machine learning to intrusion detection, several contributions have been made to the general machine learning domain.

- Learning from imbalanced data has been identified as one of the reasons for poor detection of certain classes of intrusion. The empirical research conducted in this thesis demonstrates how commonly adopted techniques such as Base classifier and deep learning perform poorly compared to our proposed classifier. We provide some designs of ensemble classifiers which perform with the new combination of base level classifiers. We also propose a new hybrid ensemble classifier which is mainly combination of two ensemble classifier.

- All the features on a dataset are not equally important for training and testing a classifier. We provide a new set of features according to their importance to improve the detection rate of the ensemble classifiers.

- The combination of base classifiers had not been considered previously in the literature. Addressing this is an important contribution to intrusion detection research. Furthermore, a novel approach to evolving ensembles classifier has been proposed, which successfully learns from imbalanced data and offers the user a wide range of solutions that exhibit different classification. From this, the user can select the solution that gives the best performance for the particular application.

1.6 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 provides an introduction to the domain of intrusion detection, which is followed by different types of attacks, detection system and existing product in the market. Different types of machine learning classifier particularly ensemble machine learning classifier is discussed in chapter 3. Chapter 3 also considers classifier combination, which is widely employed in recent literature to improve upon the performance of single classifiers. Chapter 4 discusses the dataset used by this research work and the methodology of this research work is also discussed in this chapter. Chapter 5 presents the experimental results of the algorithms. The performance of the algorithm was evaluated on independent training and testing dataset and was compared with existing classifiers. Chapter 6 concludes the thesis and offers suggestions for further work.

1.7 Summary

This thesis considers the application of machine learning to network based intrusion detection. Section 1.2 outlines the background and motivation for the work presented in this thesis. The research objective is discussed in Section 1.3. The literature is discussed in Section 1.4, followed by the contribution of this thesis work in section 1.5. The structure of the remainder of this thesis is discussed in Section 1.6.

CHAPTER 2
Intrusion Detection

2.1 Introduction

An overview of potential intrusions to computers and computer networks is presented in this chapter. A taxonomy of intrusion detection systems is provided here. As an emerging area of research, intrusion detection in networks is discussed in this chapter.

2.2 Intrusions Detection

The high demand for usage of internet is growing rapidly and so is an increase of threats on the network. A report by Symantec from 2016 implies that they have discovered more than 430 million new malwares just in 2015, an increase of 36 percent more than the year before [43].

Attacks can be varied in a long range such as Brute Force Attack, Heartbleed Attack, DoS Attack, DDoS Attack, Web Attack etc. The bandwidth of the network is increasing rapidly as the number of users of the internet are increasing. There is a huge variation of standard speed today which is from 1Gbps to 10Gbps for an average data center. The Download speed and Upload speed is different for big tech. companies like Google, Facebook, etc., or big corporate companies, which is from 40 Gbps to 100Gbps [44] [45].

Network-based Intrusion Detection System (NIDS), is a security tool which protects from an inside attack, outside attack and unauthorized access into the network. Which is designed by software and/or hardware. The most familiar concept is firewall which is built to protect the entire network from unauthorized access by IP address and port number and managing these activities by NIDS. It has extensive and wide range working applications which includes identifying the number of intrusion attempts on the network; for example, denial of service attack, hacking, etc.

NIDS is generally placed outside the firewall where the entire external traffic can be monitored by sensing and detecting the anomaly activities. When in a complex network, for example, a device connected to 1000 nodes, due to the complexity of network, it is the best decision to opt for a NIDS to keep track of changing network environment. Which brings to a conclusion as only one IDS in any network can compromise of confidential or sensitive data.

An overwhelmed NIDS can easily become a bottleneck in a network. During this case, incoming and outgoing packets may experience long delays due to the inspection of last packets or in the worst case, NIDS can drop the packet. An attacker can take this advantage easily. For example, any intrusion cannot be detected if the dropped packet has some properties for intrusion which results an incomplete packet matching.

Furthermore, to protect the confidentiality of any network for example, an organization network structure from an attack and make sure the privacy of all users, network administrator is protected. There are large varieties of machine learning algorithms that have been widely used for detecting anomaly. For example, Artificial Neural Network (ANN), Support Vector Machine (SVM), Random Forest, Self Organized, Naive-Bayesian, and Deep learning. There has been a subsequent development of Network Intrusion Detection System as classifiers to differentiate any anomaly from normal traffic.

Our approach proposes ensemble machine learning based approach which can provide a new insight to overcome the difficulties and challenges of developing a reliable and efficient Network Intrusion Detection System. We have used the latest NSL-KDD dataset of Canadian Institute for Cybersecurity (CIC) [46]. We also provide a comparison to other techniques.

In general terms, intrusive behavior can be considered as any behavior that deviates from the normal and expected use of the system. There are many types of intrusion, which makes it difficult to give a single definition of the term. Asaka et al. [47] [48] offer the following breakdown of a successful intrusion:

Surveillance/probing stage: The intruder attempts to gather information about potential target computers by scanning for vulnerabilities in software and configurations that can be exploited. This includes password cracking.

- Activity (exploitation) stage: Once weaknesses have been identified in the previous stage, they can be exploited to obtain administrator rights to the selected host(s). This will give the intruder free access to violate the system. This stage may also include *Denial of Service (DoS)* attacks, as detailed further below.

- Mark stage: After the exploitation stage, the attacker may be free to steal information from the system, destroy data (including logs that may reveal that the attack took place), plant a virus or spyware software, or use the host as a medium for conducting further attacks. After which, this marks the stage where the attacker has achieved his or her goal(s) of the attack [47].

- Masquerading stage: In this final stage, the intruder will attempt to remove traces of the attack by, for example, deleting log entries that reveal the intrusion.

The two first stages are further refined into an attack taxonomy that is widely adopted in the literature to classify attacks into four categories: Probing, Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L) [49] [50].

2.3 Intrusion Detection System

The specific architectures of IDSs are not discussed here, as these are diverse and continue to evolve with time. In general terms, Verwoerd et al. [51] have identified 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.

2.4 Types of IDS

An IDS may be described according to different characteristics, [48]:

- Data Source: host based, or network based.
- Structure: centralized or distributed
- Detection method: signature or anomaly detection.
- Reaction: passive or active.
- Usage frequency: real-time or off-line.

There are also some other types of IDS which will be described below one by one.

For detecting abnormality of a network IDS tool is used. It monitors the behavior or pattern of the network or system and notifies the administrator regarding any kind of abnormal activities; Figure 2.1 reflects the total classification of an IDS.

Firstly, IDS can be classified on basis of input data source. NIDS monitors the pattern or behavior of whole network whereas HIDS (Host-Based IDS) mainly monitors the activities of a system or network for instance: incoming TCP connection attempts, traffic flow of the network, login information and CPU, Memory, RAM usage, etc.

In general, there are only two types of IDS: One is Signature based and another is Anomaly based.

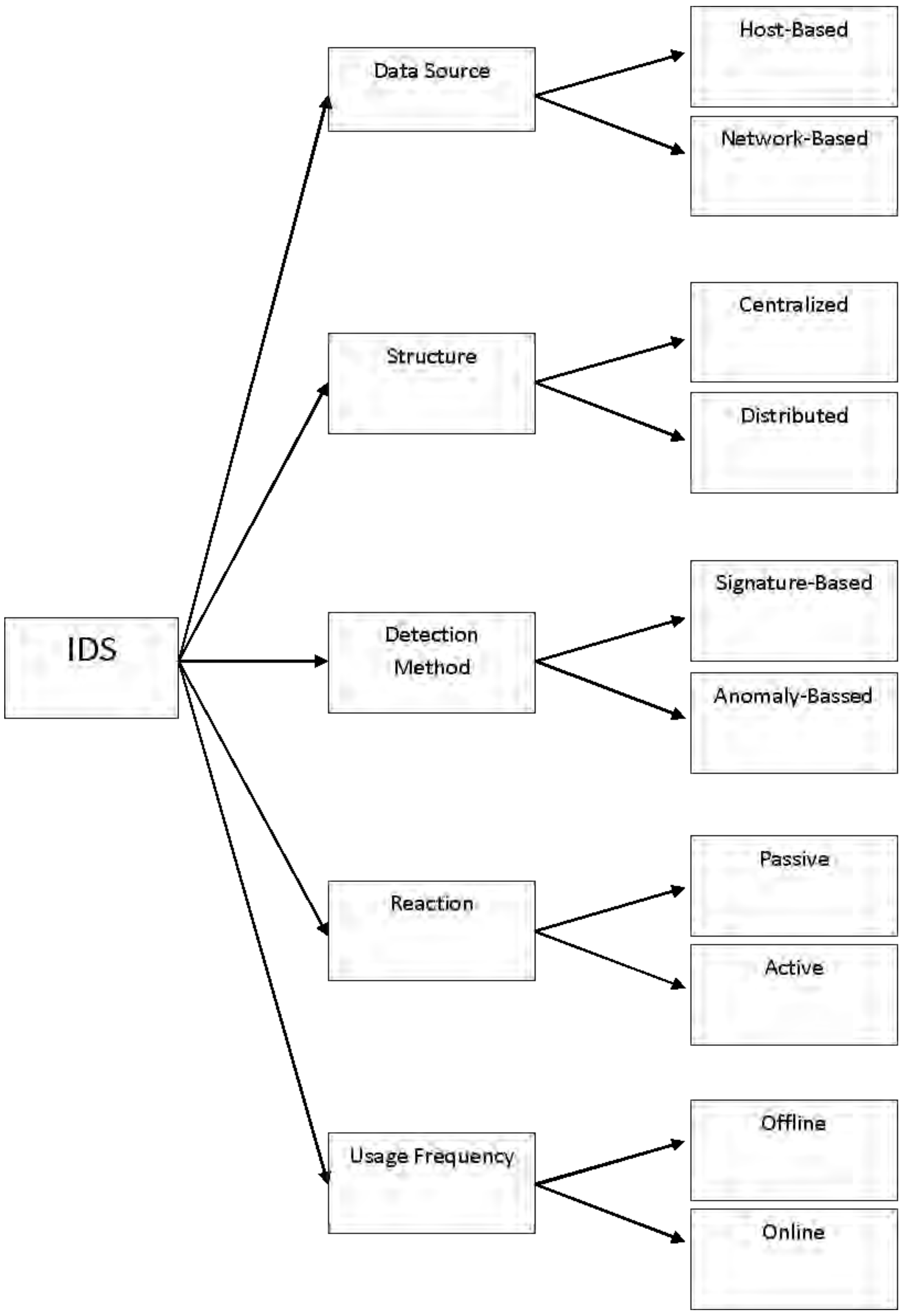


Figure 2.1: Basic types of IDS

There is another perspective of classifying IDS system which is based on reaction. In that way, there are two types of IDS system one is active, and another is passive. Active

IDS is able to take immediate action on any attack and inform the administrator. On the other hand, passive IDS stores the log details of intrusion and after that notify to the administrator.

Lastly, based on usage of frequency, we can classify an IDS system as offline IDS and Online IDS. Offline IDS is used for analyzing pre-logged data to detect any attack whereas an Online IDS use continues runtime new data to detect an attack.

Also considered here is ‘detection approach’, which describes, at a lower level, the strategies used to detect intrusions, and this is related to ‘detection method’.

2.4.1 Detection Method

The intrusion may be detected by many different techniques like signature pattern matching, anomaly detection, and many other techniques. The main two detection methods, referred to as signature detection and anomaly detection [52], [53], [54]. These terms are also known as knowledge based and behavior based intrusion detection [55], [56]. The former attempts to encode knowledge of known intrusions (misuses), typically as rules, and use this to screen events (also known as a misuse based IDS). The latter attempts to ‘learn’ the features of event patterns that constitute normal behavior, and, by observing patterns that deviate from established norms, detect when an intrusion has occurred [57]. Some IDSs offer both capabilities, typically via a hybridization of techniques [58]. However, a system may also be modeled according to both normal and intrusive data, which has become a common approach in recent research adopting machine learning techniques [59-63].

However, the performance of Signature based NIDS starts to decrease and becomes challenging whenever there is any unknown attack or when there is an anomaly traffic in the network. The signatures are pre-installed into the IDS system which has to be matched in order to detect any attack. For the case of Anomaly based NIDS (ADNIDS), it is possible to identify any unknown attack where this idea is widely acceptable among the research community.

However, rule/signature/misuse based 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 [52] [64]. One of the benefits of anomaly detection is the ability to detect new attacks, since the system is modeled according to normal behavior.

Anomaly detection is an intrusion detection technique in which normal network behavior is captured and any abnormality in the network or malicious activity is detected. This abnormality in the network can be a sudden increase in network traffic rate (number of IP packets per second).

Signature pattern matching is an intrusion detection method where the network data is compared and analyzed with the known attack techniques that are saved in a database. For example, an Intrusion Detection System which monitors web servers might be programmed to have an attention. When an intrusion is detected and it alerts the system administrator about the details of the intrusion,

The uses cases for IDS as below:

- User: User sends a request to server and server responds by providing the requested service.
- Network: In a network, IP packets are carried from the source to destination.
- IDS: An Intrusion Detection System catches the packets from the network, analyses the packets.
- System Administrator: System Administrator is alerted by the IDS of any suspicious activity or whenever an intrusion is detected.

Uses cases description for IDS as below:

1) IP Packet: A Network gives the IP Packets to Intrusion Detection System which further processes these packets.
2) Anomaly Detection: If an Intrusion Detection System detects any abnormality in the network traffic such as changes in traffic volume, bandwidth, traffic pattern, etc. Then, it triggers the alert system.

3) Signature recognition: An Intrusion Detection System monitors and examines the traffic looking for well-known attack patterns or unknown attack, which are saved in the database and if a match is found, it triggers the alert system.
4) Alert System: It alerts the system administrator, whenever triggered by anomaly detection or signature recognition.

In the Pattern or Signature detection type IDS, it has all information about the attack to detect them [65]. On the other hand, anomaly detection type IDS has all log of normal activity of network traffic and it detects any deviation from normal activity. When it finds any abnormal activity, it treats it as attack [66]. Another type of IDS is frequency detection technique which has a fixed threshold and checks whether anything crosses the threshold.

2.5 Physical Network

There are several of methods in which the security of a system can be compromised. While physically compromising a computer is an important security threat, we will focus on the problem of detecting intrusions across the network. All data come to a network as packet; our aim is to analyses all parts of that data to determine whether any attack is in progress or not. Intrusion detection often considered attacks from outside network or external network and the term misuse is then assigned to describe attacks from the internal network [67]. In order to understand what type of data is useful for detecting intrusions, it is very important to acknowledge the basic route that both appropriate and malicious users may traverse to use a specific system. There are three fundamental methods of accessing to a computer which are physical access to the host computer, using physical network, and a wireless network. This thesis will focus on using a physical network, which is described below.

In physical network, data is transmitting in the form of packets through any physical media unlike wireless communication. A unit of data is called packet and packet routed from host to destination on internet or any kind of packet-switched network. The packets carry the data in the protocols. Transmission Control Protocol/Internet Protocol (TCP/IP). Each packet contains part of the body of the message. A normal packet contains 1000 to 1400 bytes. A packet consists of three parts which are header, payload, and trailer [68]. The header is the beginning of the data and consists of information about the destination. The trailer is the end of the packet and payload is the actual information. Internet protocol (IP) is used to connect users on the Internet through an IP address.

2.5.1 Packet Header Information

Figure 2.2 shows the construction of a network packet and Figure 2.3 shows the header information of IP and TCP which is situated into the network packet.

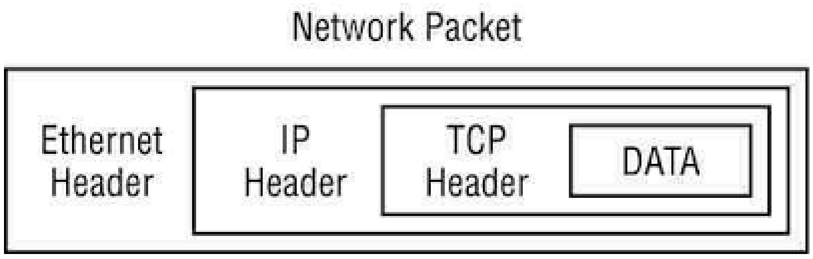


Figure 2.2: Network packet

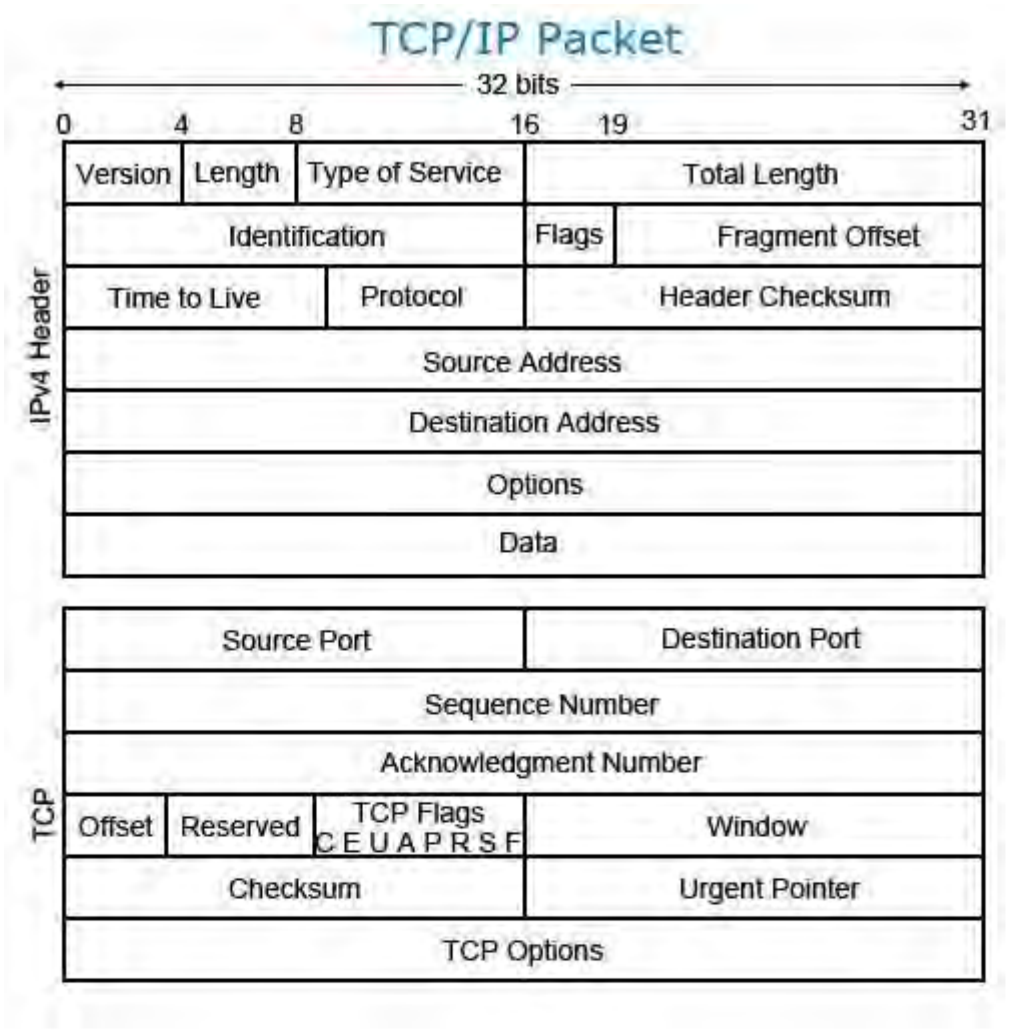


Figure 2.3: Packet header information

2.6 Network Attacks

The attack types are very huge according to number and variety. For example, any intruder may get or guess the user's password or may also monitor the traffic and after analyzing the traffic pattern, they may launch an attack. Sometimes intruder may set up an unauthorized program into the system that they managed access to that network.

Furthermore, attacker also steals information and tries to make denial of service which make the system to become zombie. Various types of attack will be explained in this section.

2.6.1 Buffer Overflow

The attack which exploits code that writing data to a buffer, swamp the boundary of the buffers also overwrites the memory location. This may cause confidentiality of sensitive data of an organization. For instance, the pointer of the instruction on stack may be overwritten in a way that the intruder wishes to execute.

2.6.2 Teardrop

This is one type of denial of service attack. In this attack, attacker exploits incorrect handling of overlapping packets. It is considered as a DoS attack because it crashes any vulnerable machine. There are some reports of being attacked by this attack on older windows or Linux operating system.

2.6.3 Ping of Death

Ping of Death attack is also one kind of Denial of service attack that uses unformatted or improperly formatted ping. When the octets of the ping is greater than 65535, this attack may happen. Linux, Windows OS are vulnerable for this attack.

Attack varies across different OS. We will discuss here for Linux and Windows operating system. The cause of variation of attacks depends of several factors. Every operating system handling their incoming packets in their way with their default rules in the network.

All packets do not maintain order while arriving. To deal with it and avoid miscommunication, packets from the same connection has given a number by the sender and rearrange by the receiving computer after arriving according to the rule of OS. The main problem is that, different receivers arrange their incoming packet differently. For instance, in the overlapping packet, if two arriving packet has the same SEQ number, they will face packet overlapping. In this situation, operating system must take decision on how to handle packets that are overlapped. The default rule for Linux is consider new packet where the default rule for windows is consider old packets [103].

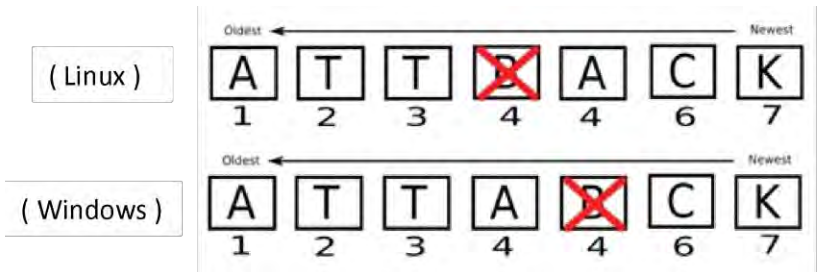


Figure 2.4: Overlapping Packet [42]

In Figure 2.4, in number 4, slot there are two packets arrived at a time and one packet is denied. Here, it shows Linux and Windows OS systems structure.

2.7 Types of Attack

The number of attacks has a large variety but most of them will fit into four main categories [69].

1. Probe
2. Denial of Service (DoS)
3. User to Root
4. Remote to User

2.7.1 Probe

Learning specific setup information of a computer or network is known as probe. We cannot say this is exactly an attack but with this information any attacker can launch an attack. Probe could be a sign of future attack. Many attacks mostly start from probe [70].

2.7.2 User to Root (U2R)

These attacks exploit vulnerabilities in operating systems and software to obtain root (administrator) access to the system.

2.7.3 Denial of Service (DoS)

This attack overloads the resources by sending unimportant information in the system and prevents actual users from accessing in the system [71]. Commercial application has been affected mostly by this attack. For example, Sony play station network has been affected by DDoS [72]. 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. Three types of *DoS* attacks are (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” [71].

2.7.4 Remote to User (R2U)

In this category, when an attacker wants to have user permission while he doesn’t have any permission to access that network. This type of attack then forwards to user to root attack. 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 (*i.e.*, obtaining data by tricking a human operator, rather than targeting software flaws [49]).
The four classes above may be used in IDS for classifying intrusions, rather than only differentiating between ‘normal’ and ‘intrusion’.

2.8 Zero Day Attack

One of our research questions consists of Zero-day attack. It can be any attack or any type of packets. From the earlier section, the attack which is not included in the first four categories is treated as the zero-day attack. This usually occurs when the time between the vulnerability found first and then exploited and the time of the application developers releases the fundamental solution to encounter the exploitation. This timeline is usually termed as the vulnerability window. These attacks can assume malware forms such as Trojan horses or worms and they are not always viruses. Updates of latest anti-malware software are often recommended, though it can only provide a minimum security against a zero day attack. The Network Based IDS(NIDS) uses raw packets as the data source and after analyzing the incoming packets it makes a pattern to decide an attack.

2.9 Open Source Intrusion Detection Systems

2.9.1 Snort

Snort is an open source software for intrusion detection and prevention system which is created by Martin Roesch in 1998. At this time, it is deployed by the sourcefire snort team [73]. This snort is single threaded which means only one job can be executed in one session without interruption. Snort uses only signature based intrusion detection from the users and the community maintains the rules like Snort VRT. The ruleset also called database [73].

First step of snort functioning is that packet acquisition. In this step, all network traffic is captured every packet is identified by structure. Snort doesn't have built-in facility, so it uses libpcap library. When data is collected, it is forwarded to next step which is preprocessor.

One type pre-processor adds another layer for complex analysis when signature based snort cannot express the rule to detect intrusion. Other type of preprocessor accepts modular plugin to view any suspicious activity into the network.
Detection Engine detects the signature and its validity according to the rule. Detection engine checks the header and payload of the arriving data to check the pattern matching and give a decision to the output. Output will set detection alert which is informed by the detection engine and show the result.

2.9.2 Rule

Snort rule is available in the databases from snort research community (VRT) and can be downloaded. They created and updated new rules for new attack on network for snort users. There may be a possibility to create any personalize signature rules for any desired packets.

Snort Sample Rule:
“alert tcp any any -> any 80 (content "|00 00 00 00|"; depth 10; msg "Bad Bytes" sid:111000111; rev:2;)”

In details of this rule, if the packet uses TCP protocol from any source address and ip address to any destination address and destination port number 80 with only four null bytes in the beginning, then alert will be triggered out. It also marked as "Bad Bytes" and with signature identification of 111000111 with revision 2.

2.9.3 Bro

Bro IDS is a passive open source analyzer. It monitors all incoming traffic looking for any suspicious activity. Normally, Bro supports a wide range of traffic analysis even it analyses outside of security domain including troubleshooting and performance measurement [74].

2.10 Commercial Intrusion Detection Systems

2.10.1 NetProwler

NetProwler is a network-based intrusion detection system by Symantec. This is a Network based IDS designed by Symantac. It uses distributed architecture and it consists of three parts

1.Agent

2.Manager

3.Console

2.10.2 NetRanger

This is designed by CISCO and also known as CISCO Netranger. It comes with all suite of software and hardware installation information and can easily set up into the network. Its maintenance and upgradation has been done by CISCO [75].

2.10.3 CFEngine 2

In 1998, Burgess wrote "Computer Immunology", a paper at the USENIX/LISA98 conference [5]. It laid out a manifesto for creating self-healing systems, reiterated a few years later by IBM in their form of Autonomic Computing. This started a research effort which led to a major re-write, CFEngine 2, which added features for machine learning, anomaly detection and secure communications.

2.10.4 Hogzilla

Hogzilla is an open source Intrusion Detection System (IDS) supported by Snort, SFlows, GrayLog, Apache Spark, HBase and libnDPI, which provides Network Anomaly Detection. Hogzilla also gives visibility of the network.

2.11 Summary

In this chapter, we have discussed regarding Intrusion Detection System (IDS) and also its different types. Packed header information discussed in section 2.5.2. In section 2.7, we discussed regarding various types of network attacks. We present a brief overview regarding open source IDS in section 2.9 whereas the commercial IDS discussed in section 2.10.

Chapter 3
Classifiers

3.1 Introduction

Machine learning, nowadays, is popular in various fields of computer science [76-79]. Because of new computing technologies, machine learning today is not like machine learning of the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks; researchers interested in artificial intelligence wanted to see if computers could learn from data. Ensemble machine learning is one of the effective types in this field which can combine the base classifiers.

3.2 Machine Learning

Machine learning needs to provide a huge amount of data for training the model where to predict the future aspects. When the model learns from the data perfectly, there is a high probability to predict the future correctly. Machine learning techniques are normally used when any problem cannot be solved by any mathematical calculation or writing any script alone. There are two categories of machine learning problems that can be addressed: one is supervised learning and other is unsupervised learning.

3.2.1 Applications of Machine Learning

While many machine learning algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to big data – over and over, faster and faster – is a recent development: [76-80]. Here are a few application areas of machine learning people are familiar with:

1. ***Data Security*** - predict malware
2. ***Personal Security*** - spot things human screeners might miss
3. ***Financial Trading*** - predict what the stock markets will do
4. ***Healthcare*** - spot cancers sooner than they are officially diagnosed
5. ***Marketing Personalization*** - lead consumers reliably towards a sale

6. ***Fraud Detection*** - PayPal is using machine learning to fight money laundering

All of these things mean that it is possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, more accurate results, even on a very large scale. And by building precise models, an organization has a better chance of identifying profitable opportunities or avoiding unknown risks. Most industries working with large amounts of data have recognized the value of machine learning technology. By gleaning insights from this data, often in real time, organizations are able to work more efficiently or gain an advantage over competitors. For example: Financial services, Health care, Oil and gas, Government, Marketing and sales, Transportation, etc.

Some of the fields who use machine learning applications are shown in the Figure 3.1 below.

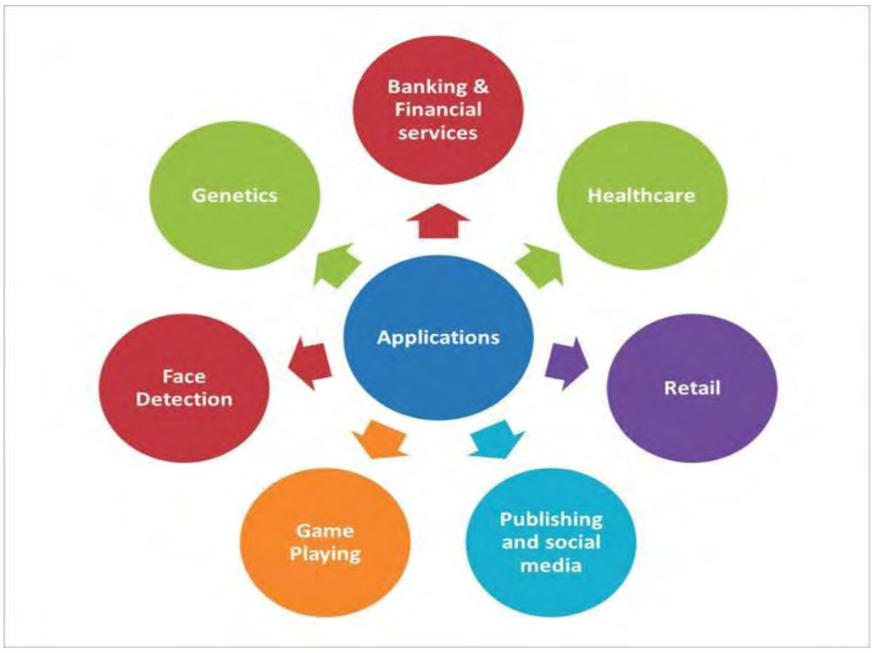


Figure 3.1: Application areas of machine learning (Source: [80])

3.3 Types of Machine Learning

Some popular types of machine learning algorithms are:

3.3.1 Supervised Learning

Supervised machine learning algorithms [81-83] are trained using labeled examples, such as an input where the desired output is known (Figure 3.2). For example, a piece of equipment could have data points labeled either "F" (failed) or "R" (runs). The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly. Through methods like classification, regression, prediction and gradient boosting, supervised learning uses patterns to predict the values of the label on additional unlabeled data.

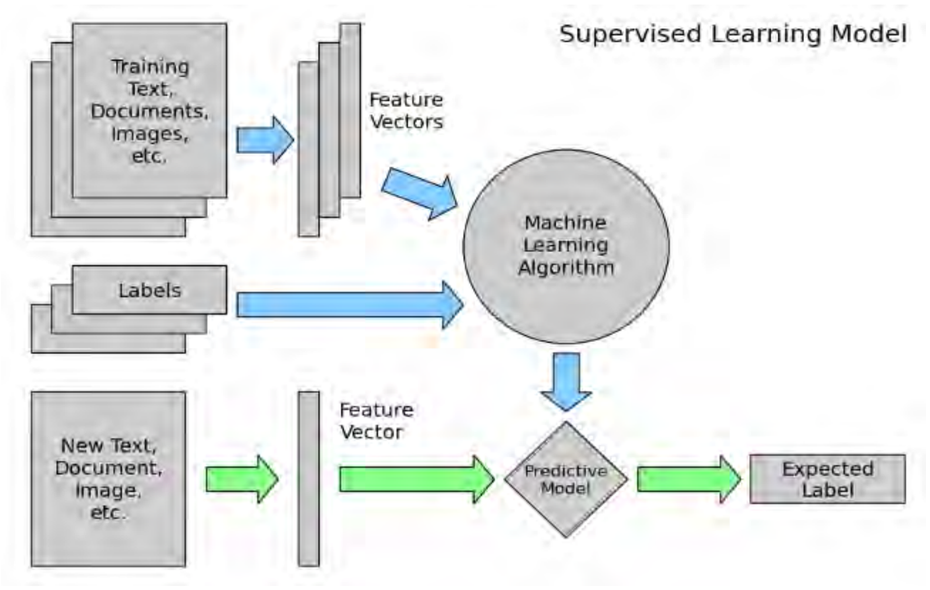

Supervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
Labels
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Expected Label

Figure 3.2: Block diagram of supervised machine learning algorithm (Source: [84])

Supervised learning is commonly used in applications where historical data predicts likely future events. For example, it can anticipate when credit card transactions are likely to be fraudulent or which insurance customer is likely to file a claim. In supervised learning, predefined dataset has been provided before training the algorithms. Firstly, these datasets are labeled and based on the labels or tags, the algorithms learn. After learning from the dataset, model can predict any future expectations [85].

3.3.2 Unsupervised Learning

Unsupervised machine learning [81-83] is used against data that has no historical labels (Figure 3.3). The system is not told the "right answer." The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within. Unsupervised learning works well on transactional data. For example, it can identify segments of customers with similar attributes who can then be treated similarly in marketing campaigns. Or, it can find the main attributes that separate customer segments from each other.

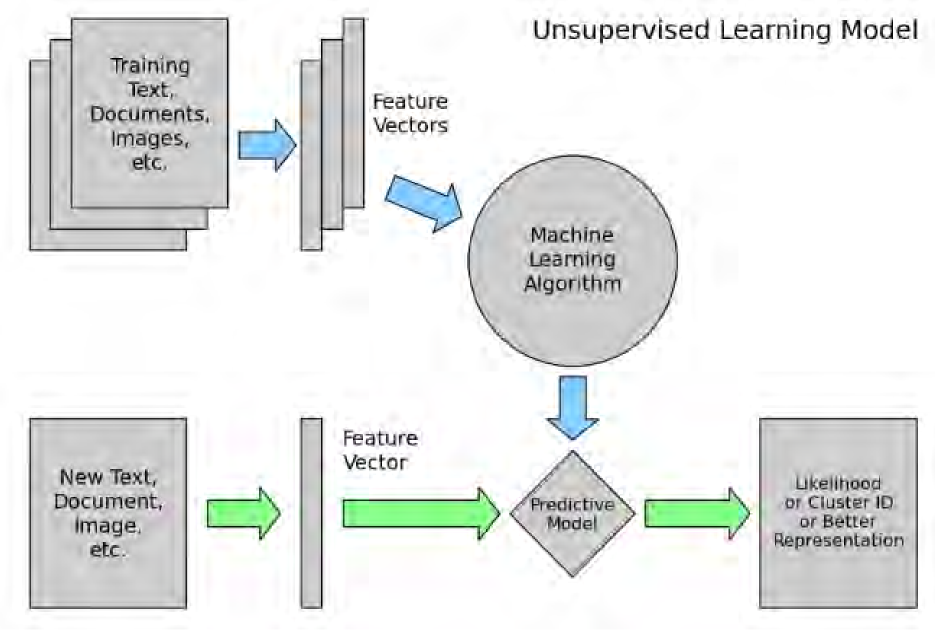

Unsupervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Likelihood or Cluster ID or Better Representation

Figure 3.3: Block diagram of unsupervised learning algorithm (Source: [81])

Popular techniques of unsupervised learning include self-organizing maps, nearest-neighbor mapping, k-means clustering and singular value decomposition. These algorithms are also used to segment text topics, recommend items and identify data outliers.

In this thesis, we propose a supervised NIDS with ensemble learning algorithm, which is compatible with the known attack as well as an unknown attack. Which we call zero-day attack. Because based on the unknown attack in the NSL-KDD test dataset, which doesn't need any experience, sees every attack, i.e., known attack or unknown attack as a new attack. Our proposed model has the capability to detect various types of new attacks, for example, DoS, Heartbleed, port scanning or any other types of attack which may cause a huge amount of network traffic. Within that time, pattern or behavior of network traffic has been analyzed by the intruder which may cause an attack. Based on previous research on this, NIDS needs to improve the detection rate particularly R2L and U2L attack.

3.3.3 Semi Supervised Learning

Semi supervised learning [81] [82] is used for the same applications as supervised learning. But it uses both labeled and unlabeled data for training (Figure 3.4). Typically it uses a small amount of labeled data with a large amount of unlabeled data (because unlabeled data is less expensive and takes less effort to acquire).This type of learning can be used with methods such as classification, regression and prediction.

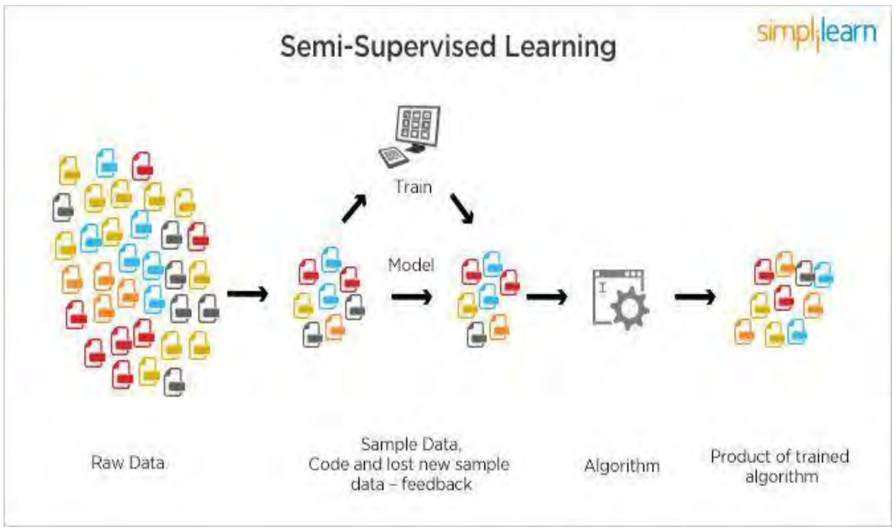

Semi-Supervised Learning
simplilearn
Train
Model
Raw Data
Sample Data, Code and lost new sample data - feedback
Algorithm
Product of trained algorithm

Figure 3.4: Block diagram of semi-supervised learning algorithm (Source:[82])

Semi supervised learning is useful when the cost associated with labeling is too high to allow for a fully labeled training process. Early examples of this include identifying a person's face on a web cam.

3.3.4 Reinforcement Learning

Reinforcement learning [81-83] is often used for robotics, gaming and navigation. With reinforcement learning, the algorithm discovers through trial and error which actions yield the greatest rewards (Figure 3.5).

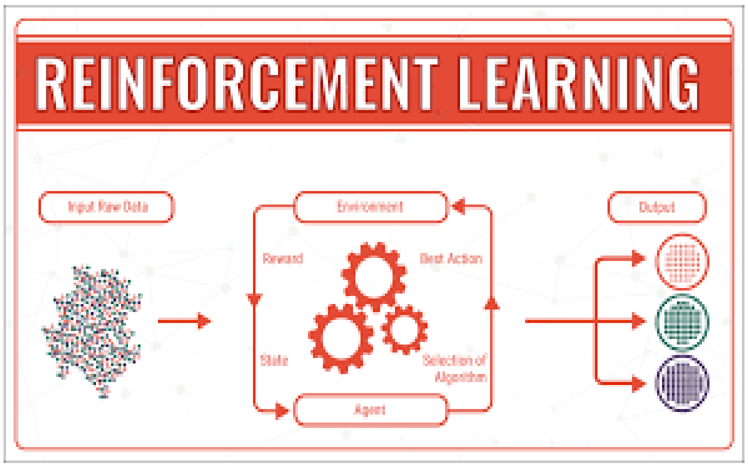


Figure 3.5: Block diagram of reinforcement learning algorithm (Source: [83])

This type of learning has three primary components: the agent (the learner or decision maker), the environment (everything the agent interacts with) and actions (what the agent can do). The objective is for the agent to choose actions that maximize the expected reward over a given amount of time. The agent will reach the goal much faster by following a good policy. So, the goal in reinforcement learning is to learn the best policy.

3.4 Base Level Classifier

In machine learning, there are different base classifier which can be used to work with ensemble machine learning. Some of the base classifiers are described below

3.4.1 Support Vector Machine (SVM)

Support Vector Machines [86] [87] are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in Figure 3.6. In this example, the objects belong either to class GREEN or RED. The separating line defines a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any new object (white circle) falling to the right will be labeled/classified, as GREEN (or classified as RED if it falls to the left of the separating line).

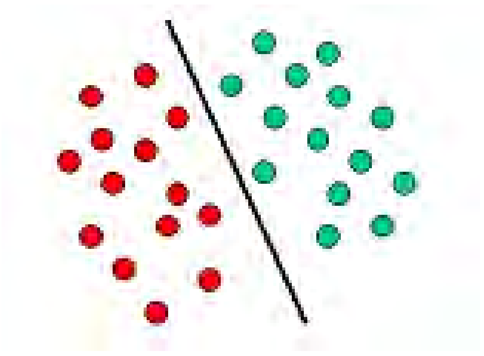


Figure 3.6: Linear decision plane in SVM (Source: [87])

Figure 3.7 is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in Figure 3.8. Compared to the previous schematic, it is clear from Figure 3.8 that a full separation of the GREEN and RED objects would require a curve which is more complex than a line. Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.

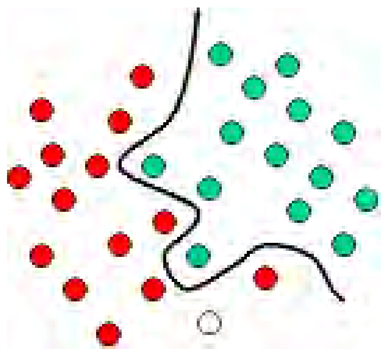


Figure 3.7: Non-linear decision plane in SVM (Source: [87])

Figure 3.8 shows the basic idea behind Support Vector Machines. Here, the original objects (left side of the schematic) are mapped, i.e., rearranged, using a set of mathematical functions, known as kernels. The process of rearranging the objects is known as mapping (transformation). Note that in this new setting, the mapped objects (right side of the schematic) is linearly separable and, thus, instead of constructing the complex curve (left schematic), all one need to do is to find an optimal line that can separate the GREEN and the RED objects.

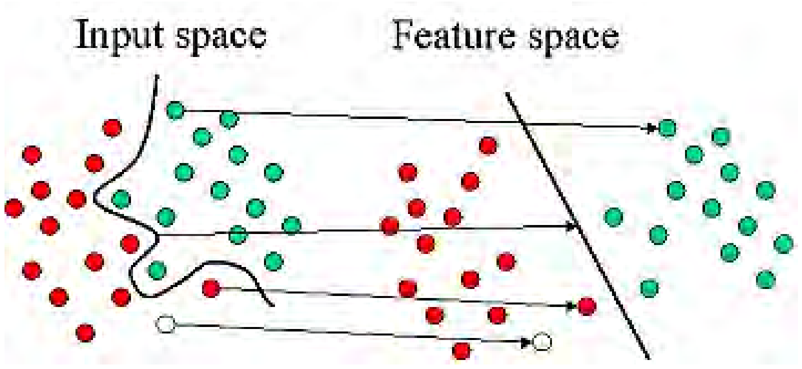


Figure 3.8: Mapping of data points from the input space to another feature space in SVM (Source: [87])

If the data is not linearly separable, then a kernel trick is used. Kernels are functions that quantify similarities between observations. Common types of kernels used to separate non-linear data are polynomial kernels, radial basis kernels, and linear kernels [88-90]. Simply, these kernels transform the data in order to pass a linear hyperplane and thus classify the data. So, the rule of thumb is to use linear SVMs (or logistic regression) for linear problems, and nonlinear kernels such as the Radial Basis Function kernel for non-linear problems. Extensions of support vector machines can be used to solve a variety of other problems, such as - multiple class SVMs using One-Versus-One Classification or One-Versus-All Classification. The chosen kernel defines the function class one is working with. The squared exponential kernel (radial basis function kernel) defines a function space that is a lot larger than that of the linear kernel or the polynomial kernel. A linear kernel allows the users to use linear functions, which are really impoverished. As the order of the polynomial kernel increases, the size of the function class increases. An n^{th} order polynomial kernel gives all analytic functions whose derivatives of order (n+1) are constant, and hence all derivatives of and above order (n+2) are zero. The RBF kernel gives access to all analytic functions (that is, all infinitely differentiable functions). So, sense the RBF kernel can be viewed as powerful as an infinite order polynomial kernel. Technically if users use squared exponential kernel, then the method is nonparametric. And if the kernel is polynomial, the model is parametric. In a way, nonparametric model means that the complexity of the model is potentially infinite, its complexity can grow with the data. If the users give it more and more data, it will be able to represent more and more complex relationships. In contrast, a parametric model's size is fixed. So, after a certain point this model will be saturated, and giving it more and more data won't help. So asymptotically assuming users have unlimited data and very weak assumptions about the problem, a nonparametric method is always better.

In the basic classification, SVM classifies the data into two categories. Given a training set of instances, labeled pairs {(x, y)}, where y is the label of instance x, SVM works by maximizing the margin to obtain the best performance in classification. More thorough descriptions can be found in [84], [91].

However, typical examples of kernels used in SVM, which have been successfully applied to a wide variety of applications, are linear, polynomials and radial basic functions.

In this study, linear functions kernel has been adopted because we believe that it is a suitable choice for our problem. We also apply other kernels (RBF and Poly) to work with this issue. The RBF kernel nonlinearly maps samples into a higher-dimensional space, Furthermore, the linear kernel is a special case of RBF as [92] shows that the linear kernel with a penalty parameter C has the same performance as the RBF kernel with some parameters. In addition, the sigmoid kernel behaves like RBF for certain parameters [93]. Moreover, the number of hyperparameters influences the complexity of model selection. The polynomial kernel has more hyperparameters than the RBF kernel. Finally, the linear kernel has less numerical difficulties.

The solution of the model parameters of SVM corresponds to a convex optimization problem.

Thus, $k(w,x) = w^T x$ is a valid kernel function. (Known as the linear kernel). We can write down as $g(x) = w^T x + b$

Polynomial Kernel Functions**:** The Polynomial kernel is defined as

$$K(x,y) = (x.y+c)^n$$

Where n is the “order” of the kernel, and c is a constant that allows to trade off the influence of the higher order and lower order terms.

Radial Basis Function (RBF)**:** Also known as a Gaussian Kernel, Radial Basis Function (RBF) kernels are often used in Computer Vision. The RBF Kernel function has the form:

$$K(x,y) = \exp \{(-||x-y||^2)/(2\sigma)^2\}$$

The term ||x-y|| is the Euclidean distance from the set of points {y}. The σ (sigma) parameter acts as a smoothing parameter that determines the influence of each of the points, y.

3.4.2 Logistic Regression

Logistic regression is firstly developed by statistician D. R. Cox in 1958 as a statistical method, and after that it is used widely in many fields [94]. In the early 1980’s, it has become routinely available in statistical packages.

Logistic regression deals with the relationship existing between a dependent variable and one or more independent variables. It provides a method for modeling a binary response variable which takes values 1 and 0. Logistic regression analysis extends the technique of multiple regression analysis to research situations in which the outcome variable is categorical. Situation involving categorical outcomes are quite common in practice. Logistic regression model has been applied in a number of contexts; which includes applications to adjust for bias, in comparing two groups in observational studies. Logistic regression analysis is part of a category of statistical model known as generalized linear models which consist of fitting a logistic regression model to an observed proportion in order to measure the relationship between the response variable and set of explanatory variables.

There are some cases where dependent variables can have more than two outcomes, are classified as multinomial logistic regression**.** The logit function (sigmoid function) is calculated as

$$P(y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

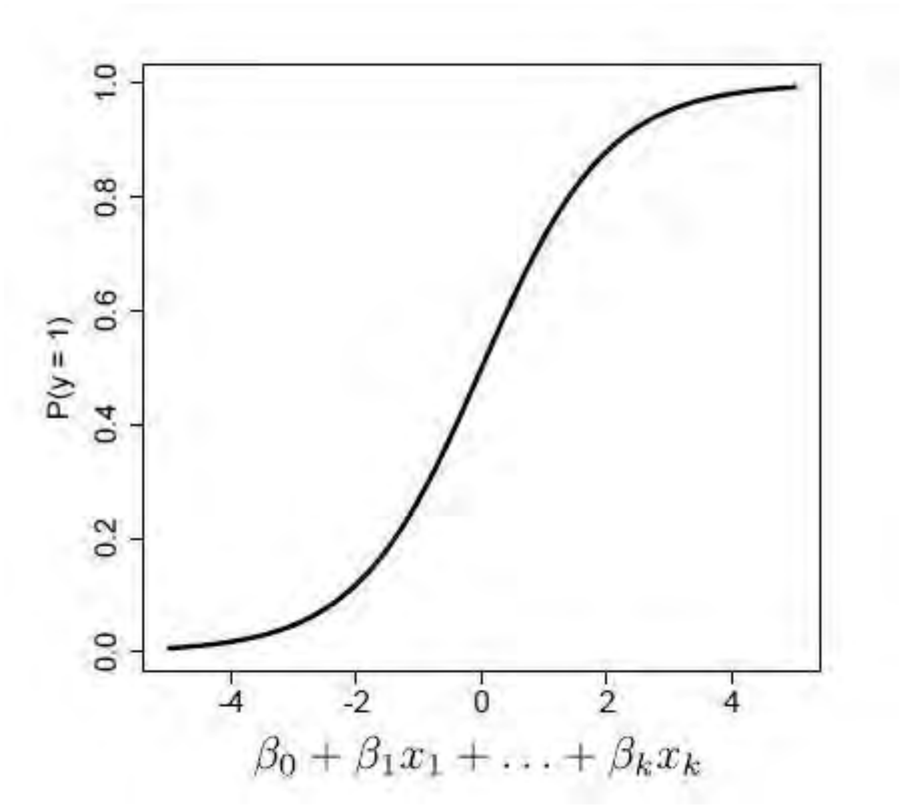


Figure 3.9: logit function

3.4.3 KNN

K nearest neighbors (KNN) [95] [96] is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point. How closely out-of-sample features resemble the training set determines how accurately the algorithm classifies a given data point [95]. When KNN is used for classification - the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

Suppose there is a dataset with n classified examples. Each classified example acts as a point in the feature space. A way to calculate the k-nearest neighbors for unclassified examples would be to find the k already classified examples that are closest to the unclassified data. Once the k neighbors have been identified, a majority class vote will take place among them to classify the new instances.

Figure 3.10 shows a graphical representation of KNN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k=3 (solid line circle), it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle.

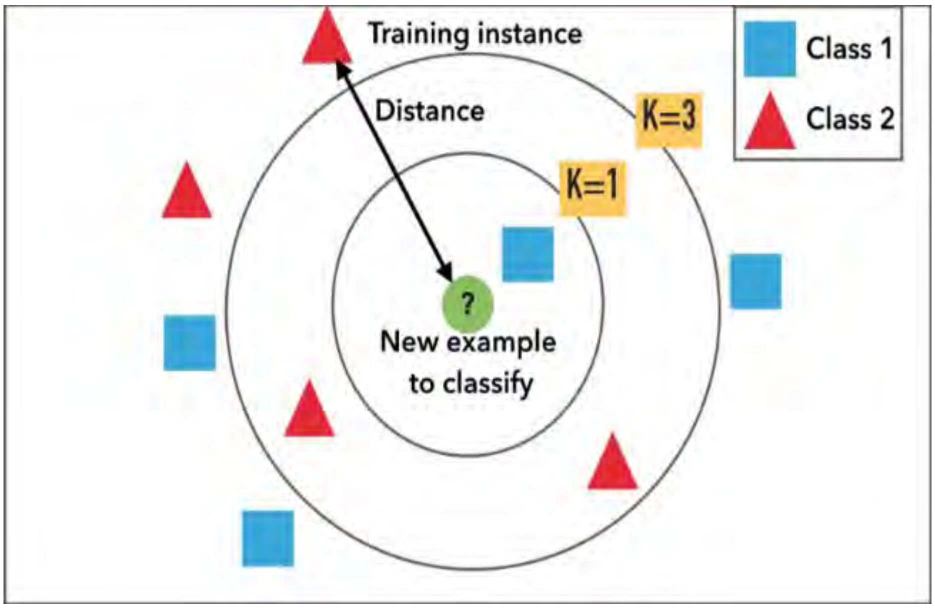


Figure 3.10: Example of KNN classification method (Source: [96])

3.5 Ensemble Machine Learning

An ensemble is itself a supervised learning algorithm, because it can be trained and then used to make predictions. Ensemble learning techniques attempt to make the performance of the predictive models better by improving their accuracy. Ensemble Learning is a process using which multiple machine learning models (such as classifiers) are strategically constructed to solve a particular problem.

An ensemble is the art of combining a diverse set of learners (individual models) together to improvise on the stability and predictive power of the model. In the above example, the way we combine all the predictions collectively will be termed as Ensemble learning.

Figure 3.11 presents a basic Ensemble structure:

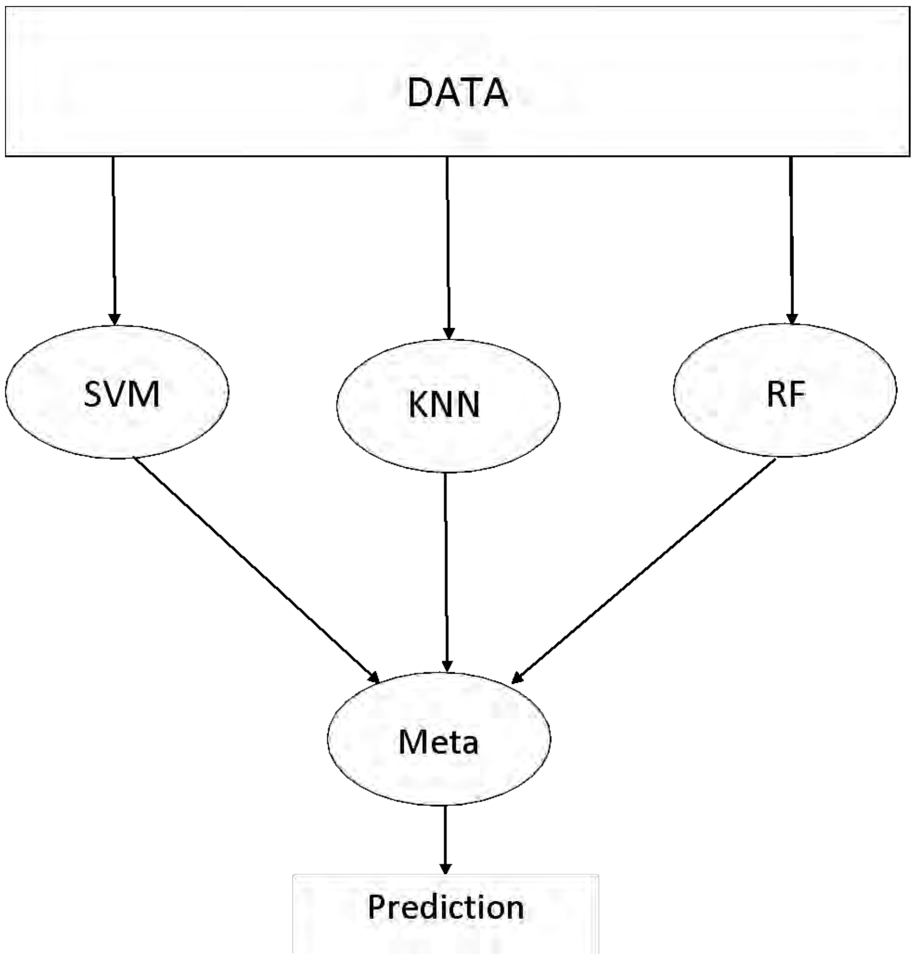


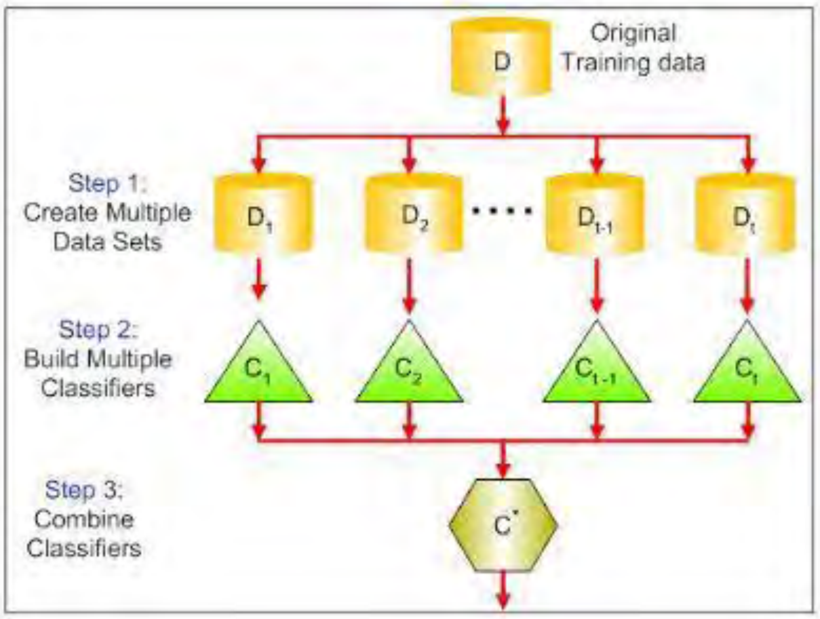
Figure 3.11: Typical prediction by ensemble learners

3.6 Types of Ensemble Learning Methods

Although there are several types of Ensemble learning methods, the following four are the most-used ones in the industry.

3.6.1 Bagging Ensemble Learning

Bagging is one of the Ensemble construction techniques which is also known as Bootstrap Aggregation. Bootstrap establishes the foundation of Bagging technique. Bootstrap is a sampling technique in which we select "n" observations out of a population of "n" observations. But the selection is entirely random, i.e., each observation can be chosen from the original population so that each observation is equally likely to be selected in each iteration of the bootstrapping process. After the bootstrapped samples are formed, separate models are trained with the bootstrapped samples. In real experiments, the bootstrapped samples are drawn from the training set, and the sub-models are tested using the testing set. The final output prediction is combined across the projections of all the sub-models. Figure 3.12 gives a brief idea of Bagging:



D
Original Training data
Step 1: Create Multiple Data Sets
D_1
D_2
D_{t-1}
D_t
Step 2: Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3: Combine Classifiers
C^*

Figure 3.12: prediction by Bagging *(Source: [97])*

Bootstrap Aggregating is an ensemble method. First, we create random samples of the training data set with replacement (sub sets of training data set). Then, we build a model (classifier or Decision tree) for each sample. Finally, results of these multiple models are combined using average or majority voting.

As each model is exposed to a different subset of data and we use their collective output at the end, so we are making sure that problem of overfitting is taken care of by not clinging too closely to our training data set. Thus, Bagging helps us to reduce the variance error.

Combinations of multiple models decrease variance, especially in the case of unstable models, and may produce a more reliable prediction than a single model.

3.6.2 Boosting Ensemble Learning

Boosting is a form of *sequential learning* technique. The algorithm works by training a model with the entire training set, and subsequent models are constructed by fitting the residual error values of the initial model. In this way, Boosting attempts to give higher weight to those observations that were poorly estimated by the previous model. Once the sequences of the models are created the predictions made by models are weighted by their accuracy scores and the results are combined to create a final estimation. Models that are typically used in Boosting technique are XGBoost (Extreme Gradient Boosting), GBM (Gradient Boosting Machine), ADABoost (Adaptive Boosting), etc.

Boosting is an iterative technique which adjusts the weight of an observation based on the last classification. If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa.

Boosting is a sequential technique in which, the first algorithm is trained on the entire data set and the subsequent algorithms are built by fitting the residuals of the first algorithm, thus giving higher weight to those observations that were poorly predicted by the previous model. It relies on creating a series of weak learners each of which might not be good for the entire data set but is good for some part of the data set. Thus, each model actually boosts the performance of the ensemble. Figure 3.13 gives a brief idea of Boosting.

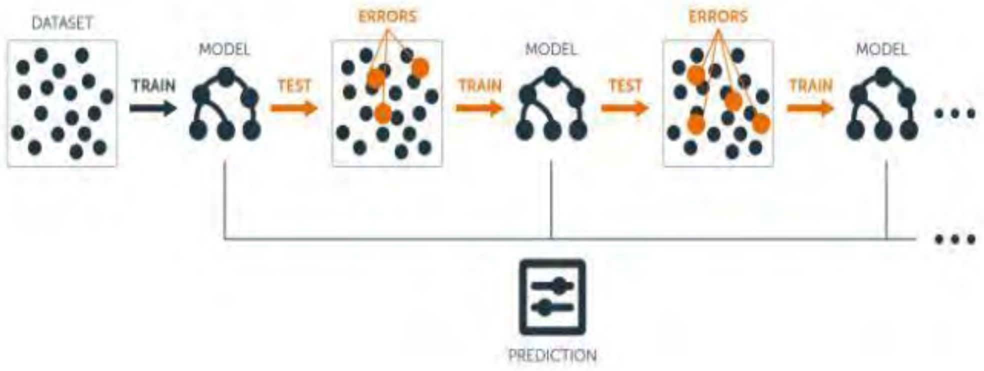


Figure 3.13: Prediction by Boosting *(Source: [98])*

3.6.3 Voting Ensemble Learning

Voting is one of the most straightforward Ensemble learning techniques in which predictions from multiple models are combined. The method starts with creating two or more separate models with the same dataset. Then, a Voting based Ensemble model can be used to wrap the previous models and aggregate the predictions of those models. After the Voting based Ensemble model is constructed, it can be used to make a prediction on new data. The predictions made by the sub-models can be assigned weights. Stacked aggregation is a technique which can be used to learn how to weigh these predictions in the best possible way. Figure 3.14 gives an idea of Voting-based Ensembles:

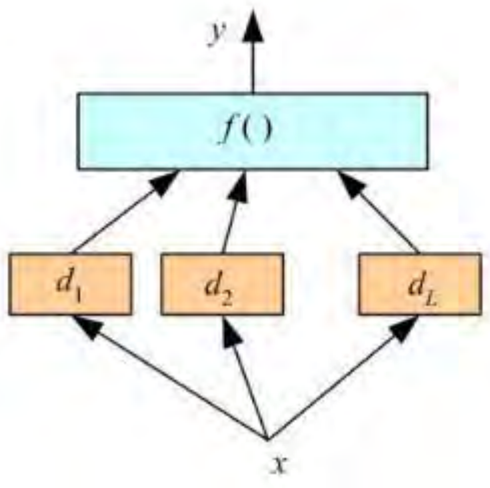


Figure 3.14: prediction by voting *(Source: [97])*

For voting classifiers, we used different combination of DT, KNN, LR SVM as base level classifiers and design 4 different voting classifiers

3.6.4 Stacking Ensemble Learning

Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features. The base level often consists of different learning algorithms and therefore stacking ensembles are often heterogeneous.

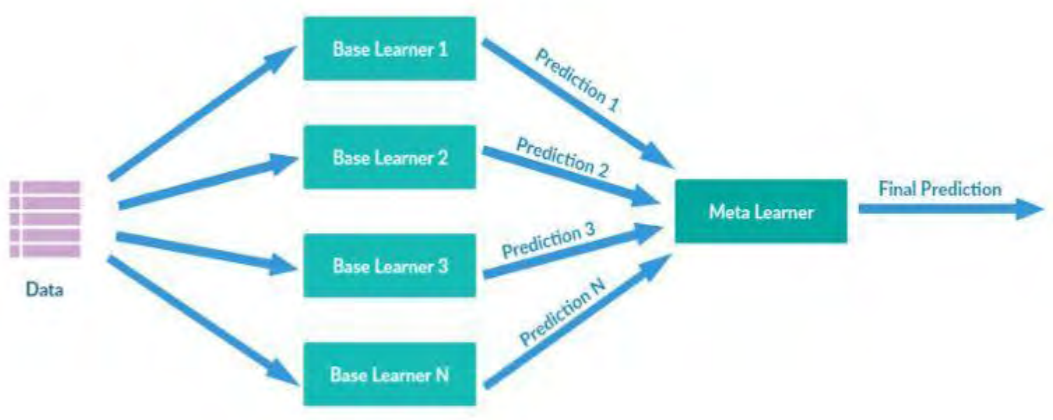


Figure 3.15: prediction by Stacking *(Source: [99])*

The ensemble of models will give better performance on the test case scenarios (unseen data) as compared to the individual models in most of the cases. The aggregate result of multiple models is always less noisy than the individual models. This leads to model stability and robustness.

3.7 Summary

In this chapter, we discussed regarding various Machine Learning Classifiers (MLC). At first we presented a brief discussion on supervised and unsupervised learning and after that we also discussed about the base machine learning classifiers. At the end of this chapter, we discussed various types of ensemble machine learning classifiers including Voting, Stacking, Bagging and Boosting classifiers in section 3.6.

Chapter 4
Proposed Methodology

4.1 Introduction

This chapter will discuss regarding the methodology of the proposed algorithm. At first, the basic block diagram will be discussed and at the end of this chapter, every step will be discussed.

The techniques of data collection, feature extraction and feature selection will also be discussed in this chapter. After selecting significant features, the final feature set has been constructed. Now, a classification algorithm needs to be developed which will maximize the detection rate. For this purpose, a number of classification algorithms are required to be tested for this dataset. Their description and working procedure will be discussed in this chapter. Their performance will be presented in the next chapter. At the end, the best classifier for this dataset will be selected for the proposed method.

A basic block diagram of the proposed algorithm is presented in Figure 4.1

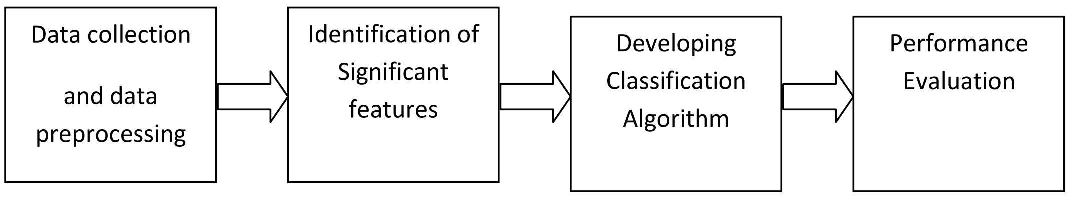

Data collection and data preprocessing
Identification of Significant features
Developing Classification Algorithm
Performance Evaluation

Figure 4.1: Basic block diagram for our workflow

4.2 Proposed Algorithm

This thesis proposes a set of new methods for intrusion detection based on network traffic. It uses the header information of a packet (IP and TCP header as discussed in section 2.5.1) extracted from the network traffic. They create an opportunity to start a new era of working with less complex features and developing newer algorithms. So, some important feature selection techniques have been incorporated here to select optimal features, e.g., F-test (filter method). Feature selection has a significant impact on subsequent stages of the learning. The selection of appropriate classification techniques is one of the most important aspects for prediction issues. To find out the appropriate one for the proposed method, several ensemble classification algorithms (Voting, Stacking, Bagging Boosting) have been experimented with and then the best one was selected. Use of cross-validation technique [49] here ensured random split of data into test set. Then, performance of the proposed algorithm has been checked. Results are discussed in chapter 6. Before discussing the proposed methodology of this research work step by step, the dataset used in this research work will be described first.

4.2.1 Dataset Description

The inherent drawbacks in the KDD cup 99 dataset [100] have been revealed by various statistical analyses has affected the detection accuracy of many IDS modeled by researchers. NSL-KDD data set [101] is a refined version of its predecessor.

It contains essential records of the complete KDD data set. There is a collection of downloadable files at the disposal for the researchers. They are listed in the Table 4.1

Table 4.1: List of NSL-KDD dataset files and their description

S.No.	**Name of the file**	**Description**
1	KDDTrain+.ARFF	The full NSL-KDD train set with binary labels in ARFF format
2	KDDTrain+.TXT	The full NSL-KDD train set including attack-type labels and difficulty level in CSV format
3	KDDTrain+_20Percent.ARFF	A 20% subset of the KDDTrain+.arff file
4	KDDTrain+_20Percent.TXT	A 20% subset of the KDDTrain+.txt file
5	KDDTest+.ARFF	The full NSL-KDD test set with binary labels in ARFF format

S.No.	Name of the file	Description
6	KDDTest+.TXT	The full NSL-KDD test set including attack-type labels and difficulty level in CSV format
7	KDDTest-21.ARFF	A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21
8	KDDTest-21.TXT	A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21

In each record, there are 41 attributes unfolding different features of the flow and a label assigned to each, either as an attack type or as normal.

The details of the attributes namely the attribute name, their description and sample data are listed in the Tables 4.2, 4.3, 4.4, 4.5. The Table 4.6 contains type information of the 41 attributes available in the NSL-KDD data set. These attribute contains data about the various 5 classes of network connection vectors and they are categorized as one normal class and four attack classes. The 4 attack classes are further grouped as DoS, Probe, R2L and U2R.

Table 4.2: basic features of each network connection vector

Attribut e No.	Attribute Name	Description	Sample Data
1	Duration	Length of time duration of the connection	0
2	Protocol_type	Protocol used in the connection	Tcp
3	Service	Destination network service used	ftp_data
4	Flag	Status of the connection – Normal or Error	SF
5	Src_bytes	Number of data bytes transferred from source to destination in single connection	491

Attribute No.	Attribute Name	Description	Sample Data
6	Dst_bytes	Number of data bytes transferred From destination to source in Single connection	0
7	Land	if source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0	0
8	Wrong_fragment	Total number of wrong fragments in this connection	0
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated	0

Table 4.3: content related features of each network connection vector

Attribute No.	Attribute Name	Description	Sample Data
10	Hot	Number of hot‟ indicators in the content such as: entering a system directory, creating programs and executing programs	0
11	Num_failed _logins	Count of failed login attempts	0
12	Logged_in	Login Status : 1 if successfully logged in; 0 otherwise	0
13	Num_compromised	Number of compromised conditions	0

Attribute No.	Attribute Name	Description	Sample Data
14	Root_shell	1 if root shell is obtained; 0 otherwise	0
15	Su_attempted	1 if ``su root'' command attempted or used; 0 otherwise	0
16	Num_root	Number of root accesses or number of operations performed as a root in the connection	0
17	Num_file_creations	Number of file creation operations in the connection	0
18	Num_shells	Number of shell prompts	0
19	Num_access_files	Number of operations on access control files	0
20	Num_outbound_cmds	Number of outbound commands in an ftp session	0
21	Is_hot_login	1 if the login belongs to the ``hot'' list i.e., root or admin; else 0	0
22	Is_guest_login	1 if the login is a ``guest'' login; 0 otherwise	0

Table 4.4: time related traffic features of each network connection vector

Attribute No.	Attribute Name	Description	Sample Data
23	Count	Number of connections to the same destination host as the current connection in the past two seconds	2
24	Srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds	2

Attribute No.	**Attribute Name**	**Description**	**Sample Data**
25	Serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)	0
26	Srv_serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)	0
27	Rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)	0
28	Srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)	0
29	Same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)	**1**
30	Diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in count (23)	0
31	Srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)	0

Table 4.5: host based traffic features in a network Connection vector

Attribute No.	**Attribute Name**	**Description**	**Sample Data**
32	Dst_host_count	Number of connections having the same destination host IP address	150
33	Dst_host_srv_count	Number of connections having the same port number	25
34	Dst_host_same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in dst_host_count (32)	0.17
35	Dst_host_diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	0.03
36	Dst_host_same_src_port_rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_c ount (33)	0.17
37	Dst_host_srv_diff_host_rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_c ount (33)	0
38	Dst_host_serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)	0

Attribute No.	**Attribute Name**	**Description**	**Sample Data**
39	Dst_host_srv_s error_rate	The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_c ount (33)	0
40	Dst_host_rerro r_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)	0.05
41	Dst_host_srv_r error_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_c ount (33)	0

The attack classes present in the NSL-KDD data set are grouped into four categories [102] [100]:

1. DOS: Denial of service is an attack category, which depletes the victim’s resources thereby making it unable to handle legitimate requests – e.g., syn flooding.

2. Probing: Surveillance and other probing attack’s objective is to gain information about the remote victim e.g., port scanning.

3. U2R: unauthorized access to local super user (root) privileges is an attack type, by which an attacker uses a normal account to login into a victim system and tries to gain root/administrator privileges by exploiting some vulnerability in the victim e.g., buffer overflow attacks.

4. R2L: unauthorized access from a remote machine, the attacker intrudes into a remote machine and gains local access of the victim machine. E.g., password guessing.

Table 4.6: attribute value type

Type	**Features**
Nominal	Protocol_type(2), Service(3), Flag(4)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21), is_guest_login(22)
Numeric	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23) srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29) diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

The specific types of attacks are classified into four major categories. Table 4.7 shows this detail.

Table 4.7: mapping of attack class with attack type

Category	**Training Set**	**Testing Set**
DoS	back, land, Neptune, pod, smurf, teardrop	apache2, back, land, mailbomb, Neptune, pod, smurf, teardrop, worm processtable, udpstorm
Probe	ipsweep, nma, portsweep, satan	ipsweep, mscan, nmap, portsweep, saint, satan
R2L	spy, warezclient, ftpwrite, guesspasswd, imap, multihop, phf, warezmaster	ftpwrite, guesspasswd, httptunnel, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, wxlock, warezmaster, xsnoop
U2R	bufferoverflow, ps, loadmodule, rootkit	bufferoverflow, ps, perl, loadmodule, sqlattack, xterm
normal	normal	normal

The Table 4.7 shows the distribution of the normal and attack records available in the various NSL-KDD datasets.

Table 4.8: details of normal and attack data in different types of NSL-KDD dataset

Attack type	**KDDTrain+**	**KDDTest+**	**KDDTrain+_20Percent**	**KDDTest-21**
DoS	45,927	7460	9234	4342
Probe	11,656	2421	2289	2402
U2R	52	67	11	200
R2L	995	2885	209	2754
Normal	67,343	9711	13,449	2152
Total	125,973	22,544	25,192	11,850

Figure 4.2 clearly exhibits the count of normal and various attack class records in the different train and test NSL KDD data sets.

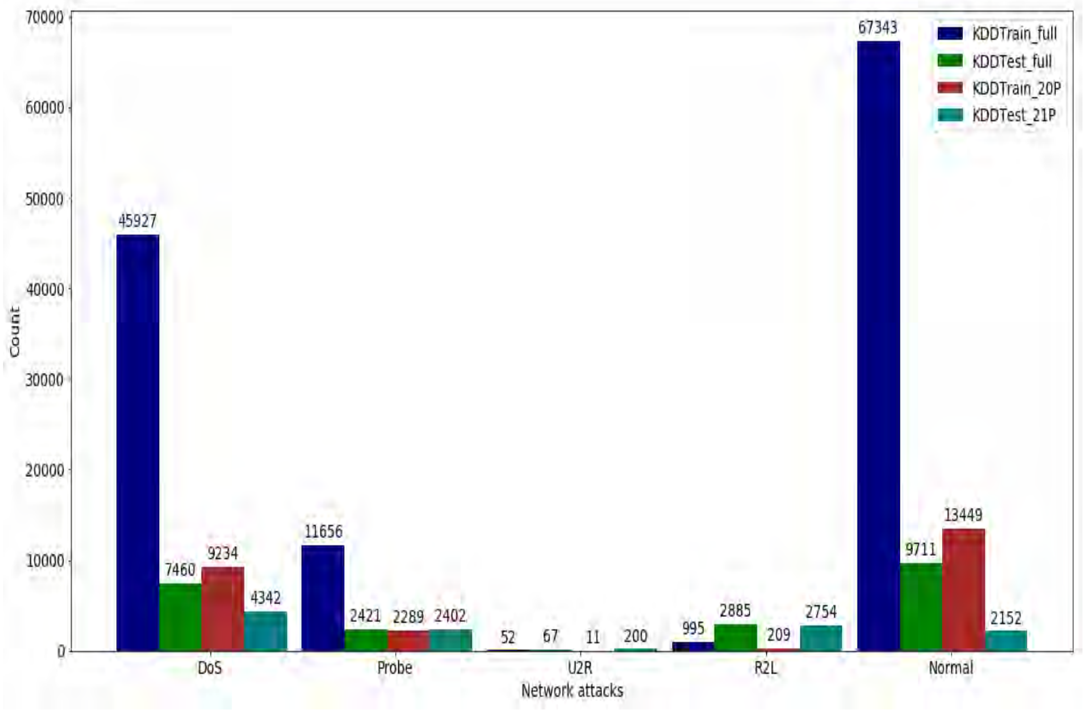


Figure 4.2: Network vector distribution in various NSL-KDD train and test data set

4.2.2 Data Extraction

The dataset which we have used in our research work was extracted from Canadian cyber crime website [46], the network traffic dataset named NSL-KDD. This dataset is an improved version by solving various lacking of KDD CUP 99 dataset. NSL-KDD dataset includes four files: KDDTrain+, KDDTest+, KDDTest-21 and KDDTest-20. There are 125,973 network traffic samples in the KDDTrain+ dataset, 22,554 network traffic samples in the KDDTest+ dataset and 11,850 network traffic samples in the KDDTest-21 dataset. There are 43 features which include 10 basic features, 12 content features, and 19 traffic features.

When the dataset is not balanced, it is difficult to classify using class label; therefore, they can be categorized into 5 network attacking groups: Normal, Probe, R2L, U2R and DoS. There are some attacks which are not present in the training set but exist in test set, which make it more realistic.

4.2.3 Data Cleaning

By using Python language programming, the first procedure or technique that been used throughout this research after extracting the dataset is data cleaning. One of the methods of data cleaning is by replacing all the attributes that only filled with categorical numeric attribute, with the name of features. By extracting the Canadian NSL-KDD dataset, we get 43 features. Among them, we remove two features which mainly contain some continuous value instead of categorical data. And, we also remove another feature which contains constant value. Finally, after cleaning the dataset, we obtained 40 useful features. Among 40 features, one feature is used as a target feature.

4.2.4 Data Labeling

One of the methods of data labeling is by replacing all the attributes that only filled with numeric indicators. Among the 40 features, there are 3 attributes which are non-numeric. They are “class”, “flag” and “protocol_type”. For instance, the feature protocol type has tcp, udp, and icmp types of attributes and after labeling, it is turned into 0, 1, and 2 respectively. Similarly, the feature “class” has 5 types of attributes and “flag” have 11 types of attributes. In the same way, all non-numerical values are transformed into numerical values after labeling and finally, our prediction target is mapped into 5 categories of classification.

4.2.5 Data Scaling

The important step after transformation is scaling. Data normalization is a process of scaling. Data scaling can avoid attributes with greater values dominating those attributes with smaller values, and also avoid numerical problems in computation. Data normalization is a process of scaling the value of each attribute into a well-proportioned range, so that the bias in favor of features with greater values is eliminated from the dataset without altering their statistical properties. Feature normalization is essential for scaling the values of each feature into a certain range (e.g., [0, 1] or any others).

The standard scores (also called z scores) of the features are calculated as follows:

$$z = (x - \mu) / \sigma$$

Where μ is the mean of the training samples, and σ is the standard deviation of the training samples.

Min-Max scaling, also referred to as normalization, consists of data being scaled to a fixed range, typically [0,1]. The only issue with this type of scaling method is that there will be smaller standard deviations, which can work to suppress the effectiveness of outliers.

In this work, Standardization is used to normalize the features during this step.

4.2.6 Feature Selection

The method that is used during this thesis work for selecting significant features is statistical significance test (F-test, filter method).

Feature Selection is essentially the process of selecting a subset of relevant and informative features from a larger collection of features that produce a better characterization of patterns belonging to different classes. Whereas principal component analysis **(**PCA**)** combines similar (correlated) attributes and creates new ones which is superior to original attributes. Feature selection technique can eliminate irrelevant and redundant features. Redundant features are those that provide no additional information beyond what is already provided by the currently selected features. Irrelevant features are those that do not provide any useful information in the given context. This has the advantage of decreasing storage requirements, reduces overfitting, improves accuracy, reduces processing time and improves the detection rate. Some examples of some filter methods include the Chi squared test, information gain and correlation coefficient scores, etc.

Filter: Filters determine the best feature subsets by using statistical approaches. Input features with a strong statistical relationship with the output feature are kept. Each feature is scored individually on certain specified criteria and the features are then ranked based on the scores and the highest ranked features are selected. Feature selection then simply becomes a manner of selecting the features based on this ranking. The methods are often univariate and consider the feature independently, or with regard to the dependent variable. Univariate feature selection works by selecting the best features based on univariate statistical tests. This feature selection examines each feature individually to determine the strength of the relationship of the feature with the response variable. These methods are simple to run and understand and are in general particularly good for gaining a better understanding of data. There are lots of different options for univariate selection.

Wrapper: Wrapper methods consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model is used to evaluate a combination of features and assign a score based on model accuracy. The search process may be methodical such as a best-first search; it may be stochastic such as a random hill-climbing algorithm, or it may use heuristics, like forward and backward passes to add and remove features.

Embedded: With embedded methods, feature selection is performed as part of the model construction process. An embedded method is usually specific to a given classification algorithm [104]. Some view embedded approaches as a type of wrapper method [105], while others view them as lying between filters and wrappers in terms of computational complexity. For instance, some embedded methods perform more efficiently than wrapper methods by directly optimizing an objective function, often defined by two or more parameters – one to encourage the goodness-of-fit and the other to penalize for a large number of variables [106] [107]. From the foregoing, we can see that each category of techniques has its own benefits and drawbacks [104] [105]. Thus, each class of techniques still has its place in the general problem of feature selection.

In this study, we focus on filter methods for identifying network attacks. Filter techniques are highly scalable (important and critical for high-dimensional datasets), relatively simple and efficient, and independent of the underlying classification algorithms [108]; Filter techniques are much faster than the other methods since they evaluate each feature once, rather than evaluating a large number of feature subsets. The filter method does not require a particular learning algorithm. It uses a heuristic to evaluate a feature subset. Filter algorithms utilize an independent measure (such as, information measures, distance measures, or consistency measures) as a criterion for estimating the relation of a set of features, while wrapper algorithms make use of particular learning algorithms to evaluate the value of features. Features filters are known to be faster than feature wrappers because heuristics are faster than induction learning. Due to the continuous growth of data dimensionality, feature selection as a pre-processing step is becoming an essential part in building intrusion detection systems.

Regarding the scoring functions, we have different functions from Scikit learn library for classification. In our research work, we have used the chi() function [109]. This function returns 2 arrays: one contains the F-Scores which are then evaluated against the chi2 distribution to obtain the p-value. If p-value is small, the parameters are said to be significant. Here, we have used the SelectKBest() class. Using this class, we can select a fixed number of significant features according to the next smallest p-value of the feature ranking.
According to the p-value, the order of 39 important features is shown in Table 4.9

Table 4.9: order of 39 important features

Number of features	**Number according to section 4.2.1**	**Feature Name**
1	22	'is_guest_login'
2	39	'dst_host_srv_serror_rate'
3	38	'dst_host_serror_rate'
4	35	'dst_host_diff_srv_rate'
5	34	'dst_host_same_srv_rate'
6	33	'dst_host_srv_count'
7	31	'srv_diff_host_rate'
8	30	'diff_srv_rate'
9	29	'same_srv_rate'
10	28	'srv_rerror_rate'
11	27	'rerror_rate'
12	26	'srv_serror_rate'

Number of features	**Number according to section 4.2.1**	**Feature Name**
13	25	'serror_rate'
14	23	'count'
15	40	'dst_host_rerror_rate'
16	41	'dst_host_srv_rerror_rate'
17	9	'urgent'
18	4	'flag'
19	12	'logged_in'
20	14	'root_shell'
21	16	'num_shells'
22	32	'dst_host_count'
23	11	'num_failed_logins'
24	36	'dst_host_same_src_port_rate'
25	21	'is_host_login'
26	37	'dst_host_srv_diff_host_rate'
27	24	'srv_count'
28	2	'protocol_type'
29	19	'num_access_files'
30	1	'duration'
31	17	'num_file_creations'
32	10	'hot'
33	16	'num_root' real'
34	13	'num_compromised'
35	6	'dst_bytes'
36	8	'wrong_fragment'
37	7	'land'
38	5	'src_bytes'
39	15	'su_attempted'

4.2.7 Selection of Classifier

Once the optimal features are sorted by their importance, these features are then taken into the classifier training stage where ensemble machine learning are employed. The classifier distinguishes attacks data from Normal traffics. Ensemble learning are categorized under the same umbrella of supervised machine learning. The main difference between classification and regression is that the output variable in regression is numerical (or continuous) while that for classification is categorical (or discrete). Common classification algorithms include logistic regression, Naïve Bayes, decision trees, and K Nearest Neighbors (KNN), Support Vector Machine (SVM), etc. Among them, we used various base classifiers as an input of ensemble learning in our work. We are selecting categorical data to do our work.

During execution of the algorithm, we used separate dataset for training and testing. After selecting the classifier, the training samples will first be fed to the developed model for learning the features of network attacks and normal network traffics. After that, the test set will be used to evaluate the trained model. The steps required for the execution of the algorithm are shown in the flowchart (Figure 4.12). To work with cross-validation, we used the full testing dataset and to work without cross validation we used the full training and testing dataset separately.

4.2.7.1 Design of Ensemble Classifier

In this section, design of classifiers namely Voting, stacking, bagging and Boosting has been discussed in detail. We design and develop our model by trial and error basis.

4.2.7.1.1 Voting

Figure 4.3 shows the design of Voting-1 classifier. In Voting-1 classifier, the base classifiers are DT, KNN and LR.

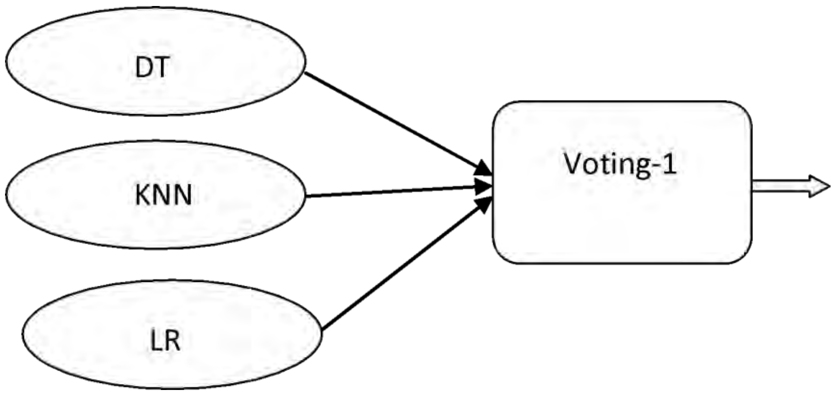


Figure 4.3: Voting-1

Figure 4.4 shows that the base classes for Voting-2 classifier is DT, SVM(kernel = poly) and LR

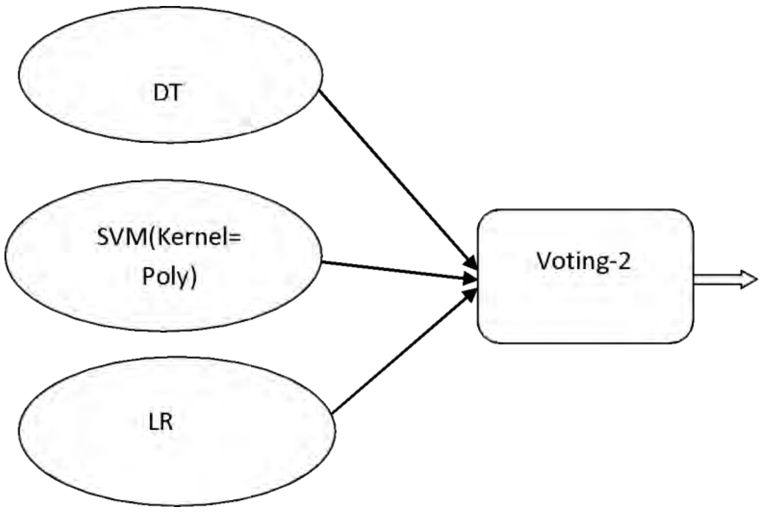


Figure 4.4: Voting-2

Figure 4.5 shows that the base class of Voting -3 is DT, SVM(kernel = rbf) and LR

The base classes of Voting-4 is DT, SVM (kernel = linear) and LR. All base classes arrangement is shown in Table 5.1

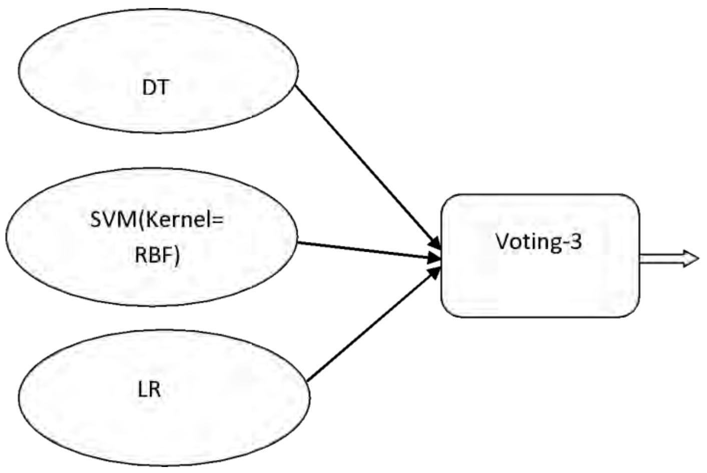


Figure 4.5: Voting-3

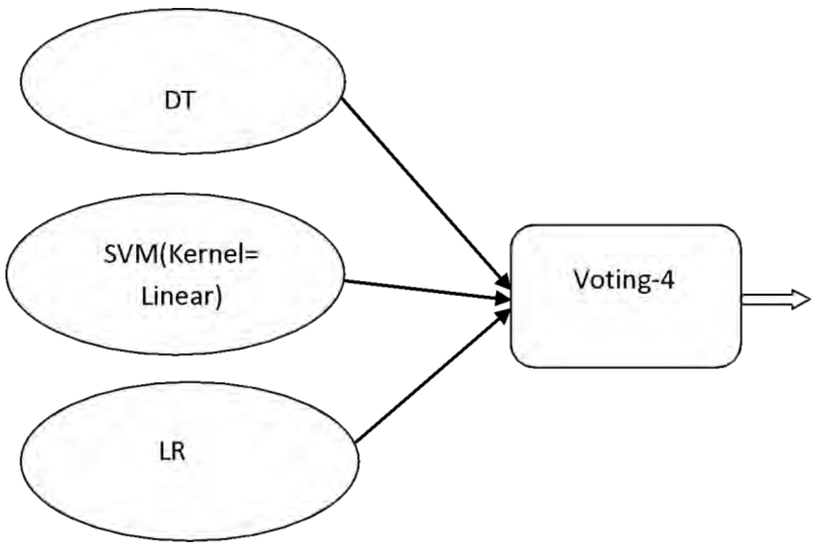


Figure 4.6: Voting -4

Voting has two types Hard Voting or Max Voting and Soft Voting. In a Hard voting system, it aggregate the predictions of each classifier and predict the class that gets the most votes. If all classifiers are able to estimate the probability of classes (predict_proba() method) then predict class with the highest class probability, averaged over individual classifiers. In our work, we used the hard voting that works on most votes.

4.2.7.1.2 Stacking

To design Stacking we need base classes as well as a meta class. Figure 4.7 shows the base classes of Stacking-1 which use SVC (kernel = linear) and Logistic regression and its meta class is also SVC (kernel = linear)

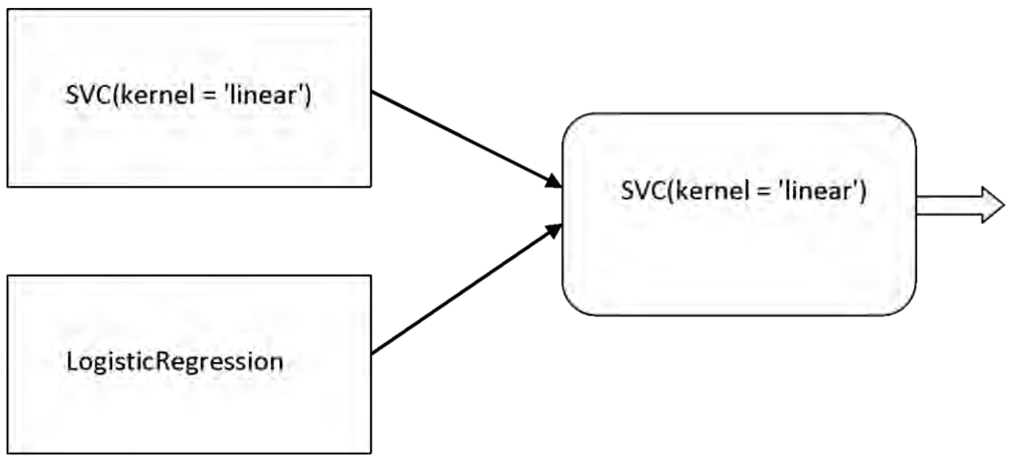


Figure 4.7: Stacking-1

Figure 4.8 shows that the base classes of Stacking-2 with SVC (kernel = rbf) and Logistic regression and its meta class is SVC (kernel = rbf)

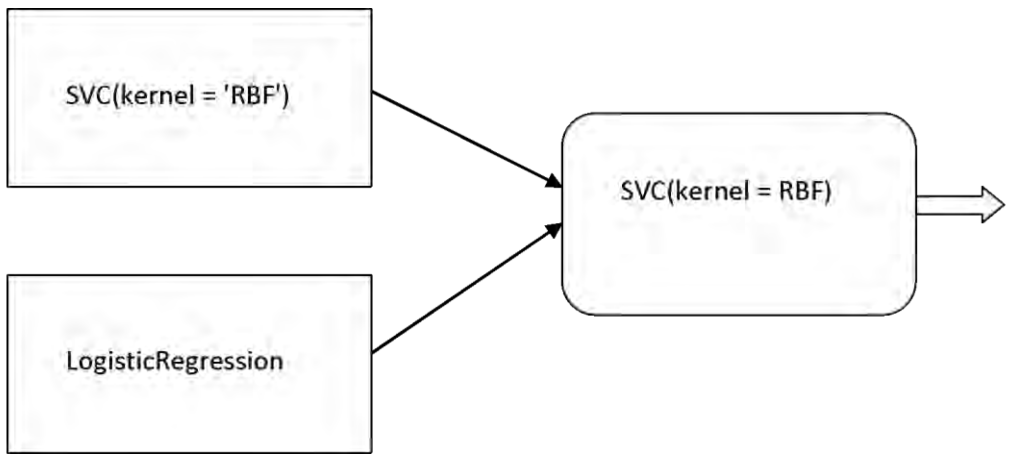


Figure 4.8: Stacking-2

From Figure 4.9 shows that the base classes of Stacking-3 with SVC (kernel = rbf) and Logistic regression and its meta class is Naïve Bayes.

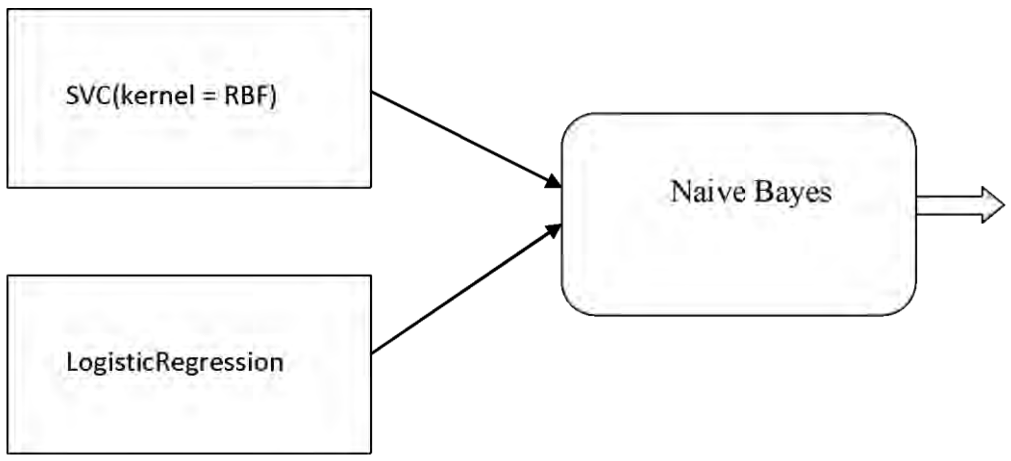


Figure 4.9: Stacking-3

From Figure 4.10 shows that the base classes of Stacking-4 with SVC (kernel = linear) and Logistic regression and its meta class is RandomForest classifier

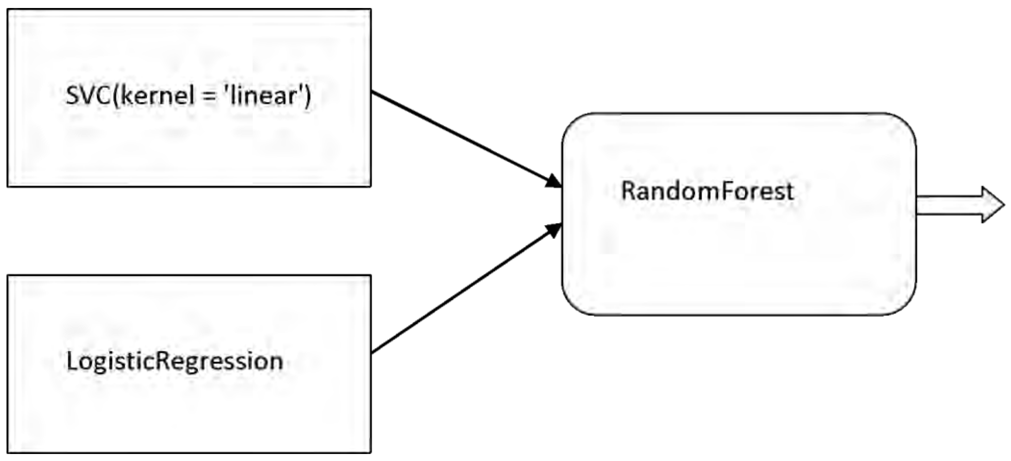


Figure 4.10: Stacking-4

4.2.7.1.3 Boosting

Boosting can take only one classifier. We designed 3 Boosting classifiers namely Boosting-1, Boosting-2 and Boosting-3 using Logistic Regression, DT and Gradient Boosting respectively. Table 4.9 shows the name of Boosting classifiers and their respective base classes.

Table 4.10: Boosting Classifiers

Classifier	Base Classifier
Boosting-1	Adaboost (LR)
Boosting -2	Adaboost (DT)
Boosting -3	GradientBoosting

4.2.7.1.4 Bagging

Bagging also can take only one classifier. We designed 3 Bagging classifiers namely Bagging -1, Bagging-2 and Bagging-3 using Logistic Regression, Random Forest and Naïve Bayes respectively.

4.2.7.1.5 Hybrid Model

We also designed two hybrid models as shown in 4.11. Hybrid Model-1 is a Voting classifier, however it takes two Boosting classifiers (AdaBoost and Gradient-Booston) as its base classifier. Figure 4.11 shows Hybrid Model-1.

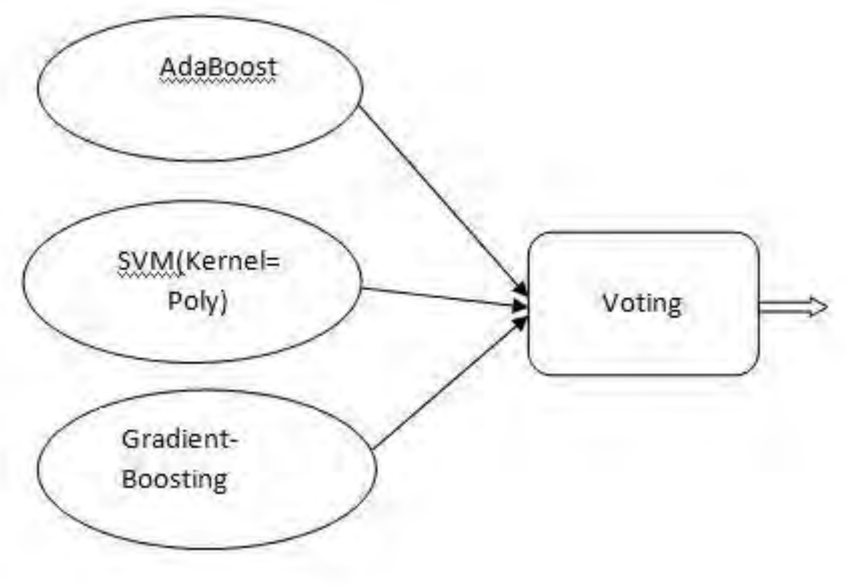
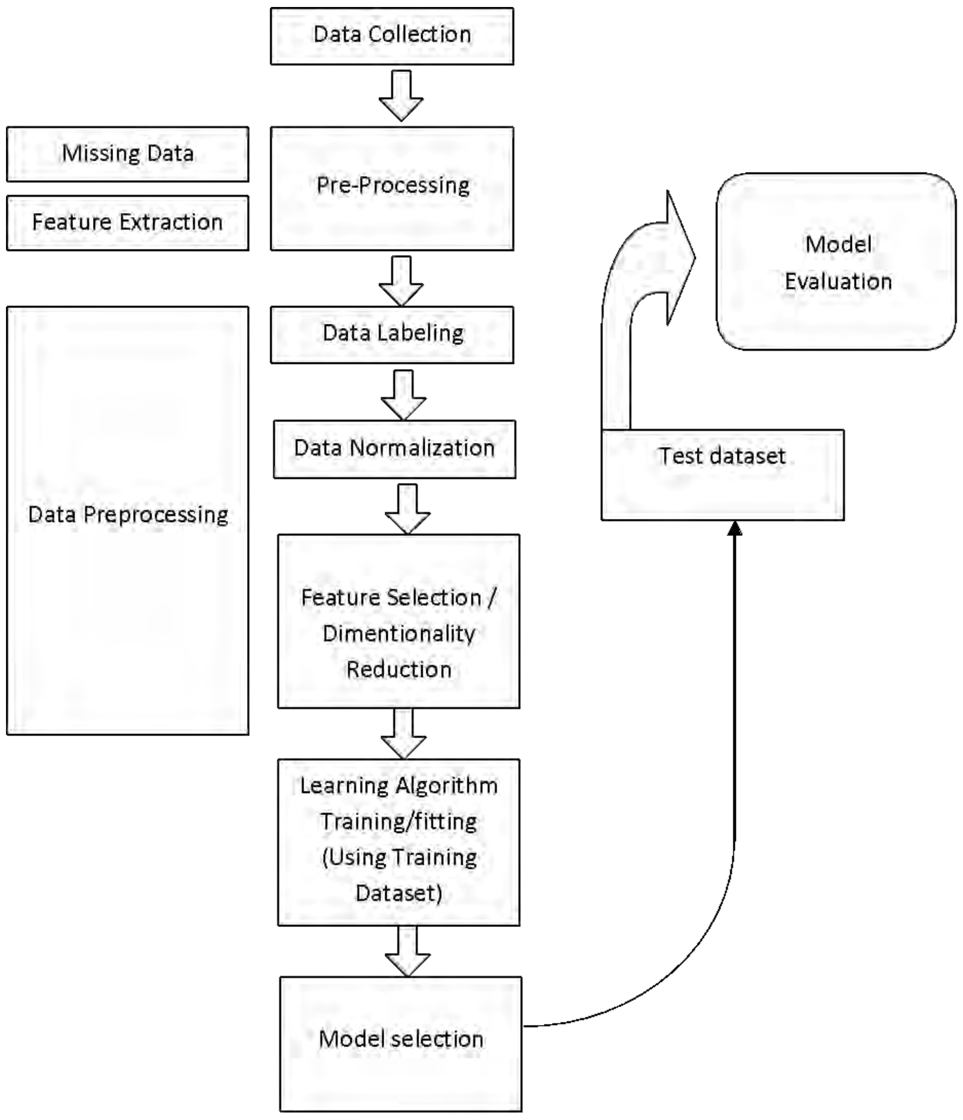


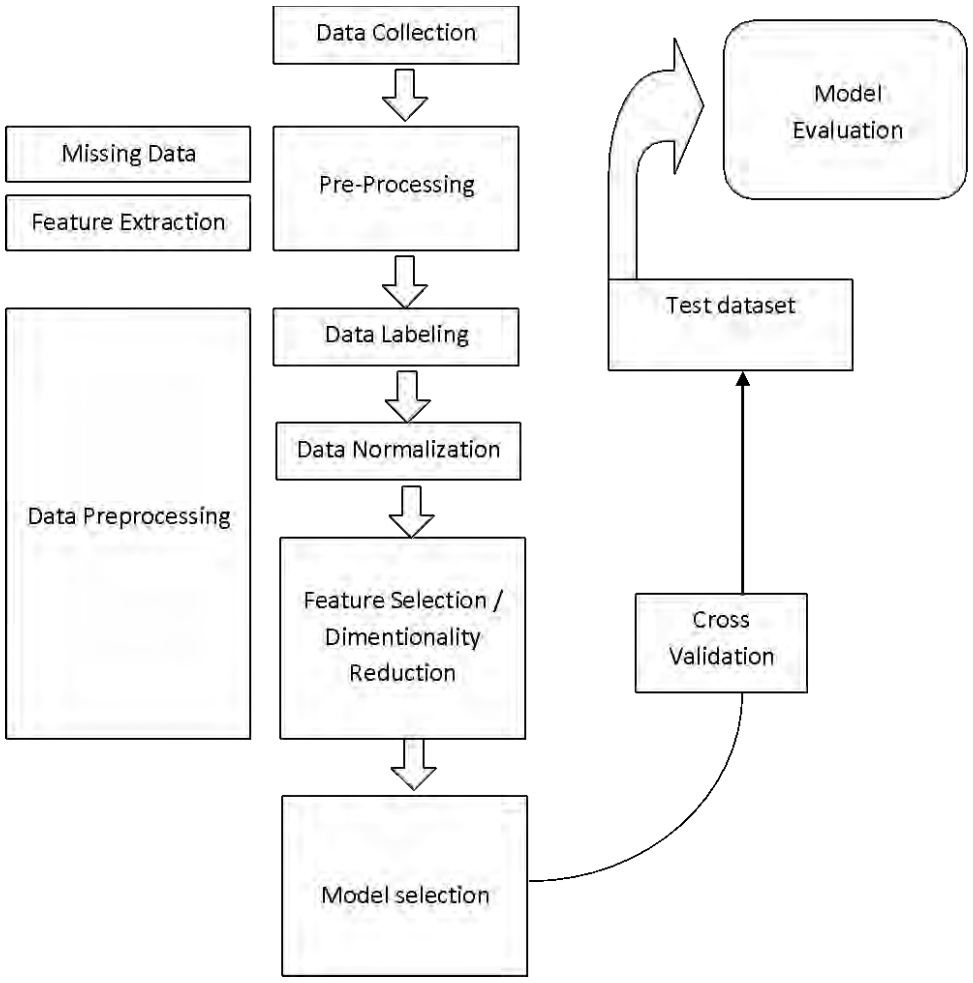
Figure 4.11: Hybrid-1

The Hybrid-2 is a Bagging classifier, and it takes voting-1 as its base classifier.

4.3 Block Diagram



(a)



(b)

Figure 4.12: Block diagram of various stages of proposed work (a) without cross-validation (b) with cross-validation

This is the workflow diagram of our research work.

4.4 Model Evaluation

For the performance evaluation in the experiment, first, we denote *TP*, *FP*, *TN* and *FN* as true positive, false positive, true negative and false negative, respectively.

Table 4.11: Specification of confusion matrix

Measures	**Specification**
P	Total number of samples classified as positive
N	Total number of samples classified as negative
True Positive (TP)	The number of samples correctly classified as attack

Measures	Specification
True Negative (TN)	Number of samples correctly classified as normal
False Positive (FP)	Number of samples wrongly classified as attack
False Negative (FN)	Number of samples wrongly classified as normal

Then, we can obtain the detection rate or sensitivity as follows:

Detection Rate (DR)

DR is indicating the proportion of actual positives values being correctly identified. It is the ratio of correctly classified network attacks to the total number of network traffic. This is given by

$$\mathrm{DR} = TP/\,(TP+FN)$$

DR is also synonymous with sensitivity. The DR measures the rate of malware samples (i.e., positive instances) correctly classified by the classification model

In intrusion detection, sometimes we pay more attention to the detection rate rather than accuracy. The higher the detection rate, the lower the probability that a network which will have the risk of attack is predicted to have no attack.

4.4.1 Confusion Matrix

In simple terms, confusion matrix is a result table that summarizes results of classification algorithm when actual true values are known.

Table 4.12: confusion matrix

Actual	Predicted	
	Attack	Normal
Attack	TP	FP
Normal	FN	TN

As the dataset has 4 types of attacks, each attack was labeled as a class. An evaluation of the proposed algorithm has been made to check the classification performance by computing the confusion matrix, considering the class labels of the training dataset as the reference and the predicted class labels as the outcome of the proposed algorithm. The output of ensemble machine learning algorithms should remain in any of the four categories in the confusion matrix: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Then, these values were used to calculate the detection rate or sensitivity of the classification.

4.4.2 ROC-AUC

In addition to the aforementioned evaluation criteria, the receiver operating characteristic (ROC) curve and the area under curve (AUC) can evaluate the pros and cons of the classier. The ROC curve shows the trade-off between the true positive rate (*TPR*) and the false positive rate (*FPR*). If the ROC curve is closer to the upper left corner of the graph, the model is better. The AUC is the area under the curve. When the area is closer to 1, the model is better.

4.4.3 Precision Recall Curve

Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.

4.5 Implementation Environment

For the implementation of this algorithm, Scikit Learn has been used. Jupiter notebook IDE has been used for executing the python [110] code.

4.6 Summary

In this research work, we used the KSL-KDD dataset. At the beginning of this chapter, we presented a brief discussion of this dataset. After that, we discussed the methodology step by step of our research work. We also discussed regarding various types of features selection technique in section 4.2.6. In section 4.2.7.1, we discussed regarding different types of new design of ensemble classifiers. At the end of this chapter, we discussed regarding model evaluation and the implementation environment of our research work.

Chapter 5
Results and Discussion

5.1 Introduction

This is the last stage and one of the most significant tasks of the thesis work where all of the data is analyzed. At first, we work without cross validation and compare the result with literature. After that we use the cross validation to train and test the dataset. When we work without cross validation, we used the full NSL-KDD train and test dataset separately. In contrast, when we use the cross validation, we used the full test dataset.

5.2 Executing the Ensemble Classifier

In this stage, we analyze and verify the data that was extracted during the experiment in-depth. We analyze the data to see if the objective set in this thesis is met. We show that how well the algorithm performed by comparing the attributes in the detection of network attacks.

5.2.1 Detection Without Cross Validation

At first, the dataset is tested without cross validation using the Voting, Bagging and Boosting ensemble machine learning classifiers.

Bagging_NB

When we have used the Bagging classifier, the base class of the bagging is Naïve Bayes. Using this Bagging_NB classifier, we calculate the detection rate according to increasing the number of feature. The number of feature is increased according to the Table 4.9. Table 5.1 shows the detection rate using Bagging_NB classifier. The Random State parameter value is 3, when we calculate the detection rate below.

Table 5.1: Detection Rate by feature importance using Bagging_NB classifier

No. of feature	**DoS(0)**	**Probe(1)**	**R2L(2)**	**U2R(3)**	**Normal(4)**
First 1 feature	0.89	0.00	0.00	0.00	0.01
First 2 features	0.89	0.00	0.00	0.04	0.10
First 3 features	0.82	0.00	0.00	0.51	0.05
First 4 features	0.82	0.00	0.00	0.55	0.05
First 5 features	0.81	0.00	0.00	0.54	0.22
First 6 features	0.81	0.00	0.00	0.60	0.44
First 7 features	0.81	0.00	0.00	0.58	0.41
First 8 features	0.81	0.00	0.00	0.63	0.42
First 9 features	0.80	0.01	0.00	0.67	0.45
First 10 features	0.79	0.00	0.00	0.72	0.55
First 11 features	0.76	0.00	0.00	0.76	0.75
First 12 features	0.75	0.00	0.00	0.78	0.79
First 13 features	0.75	0.00	0.01	0.66	0.81
First 14 features	0.75	0.01	0.01	0.66	0.80
First 15 features	0.78	0.03	0.04	0.63	0.70
First 16 features	0.78	0.06	0.06	0.60	0.69
First 17 features	0.78	0.08	0.06	0.60	0.69

No. of feature	**DoS(0)**	**Probe(1)**	**R2L(2)**	**U2R(3)**	**Normal(4)**
First 18 features	0.77	0.07	0.06	0.60	0.73
First 19 features	0.75	0.10	0.06	0.61	0.73
First 20 features	0.75	0.53	0.07	0.61	0.72
First 21 features	0.74	0.51	0.10	0.63	0.72
First 22 features	0.74	0.49	0.11	0.58	0.73
First 23 features	0.74	0.48	0.11	0.60	0.73
First 24 features	0.74	0.48	0.12	0.60	0.74
First 25 features	0.77	0.50	0.19	0.60	0.67
First 26 features	0.77	0.52	0.19	0.60	0.67
First 27 features	0.78	0.54	0.19	0.66	0.67
First 28 features	0.77	0.77	0.19	0.66	0.69
First 29 features	0.77	0.88	0.19	0.66	0.69
First 30 features	0.76	0.88	0.19	0.66	0.70
First 31 features	0.78	0.89	0.19	0.67	0.69
First 32 features	0.79	0.89	0.32	0.66	0.47
First 33 features	0.79	0.90	0.32	0.67	0.30
First 34 features	0.79	0.91	0.33	0.67	0.29
First 35 features	0.80	0.91	0.33	0.67	0.20

No. of feature	DoS(0)	Probe(1)	R2L(2)	U2R(3)	Normal(4)
First 36 features	0.82	0.91	0.33	0.60	0.16
First 37 features	0.82	0.91	0.33	0.60	0.13
First 38 features	0.79	0.91	0.39	0.61	0.10
First 39 features	0.80	0.91	0.40	0.57	0.07

We can understand the Table 5.1 more easily by graphical representation. Figure 5.1 to Figure 5.5 are the graphical representation of Table 5.1.

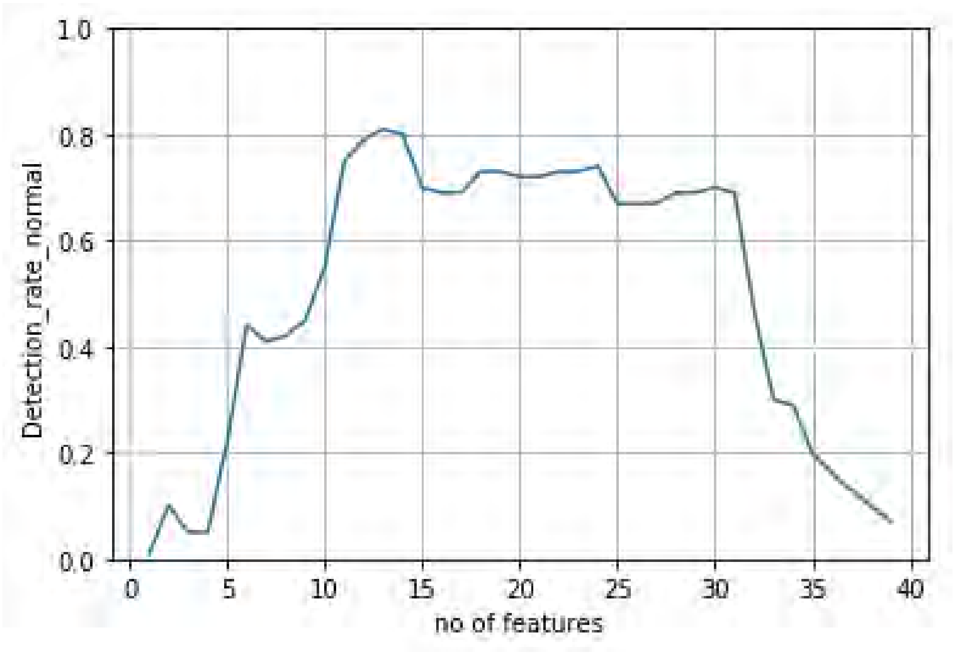
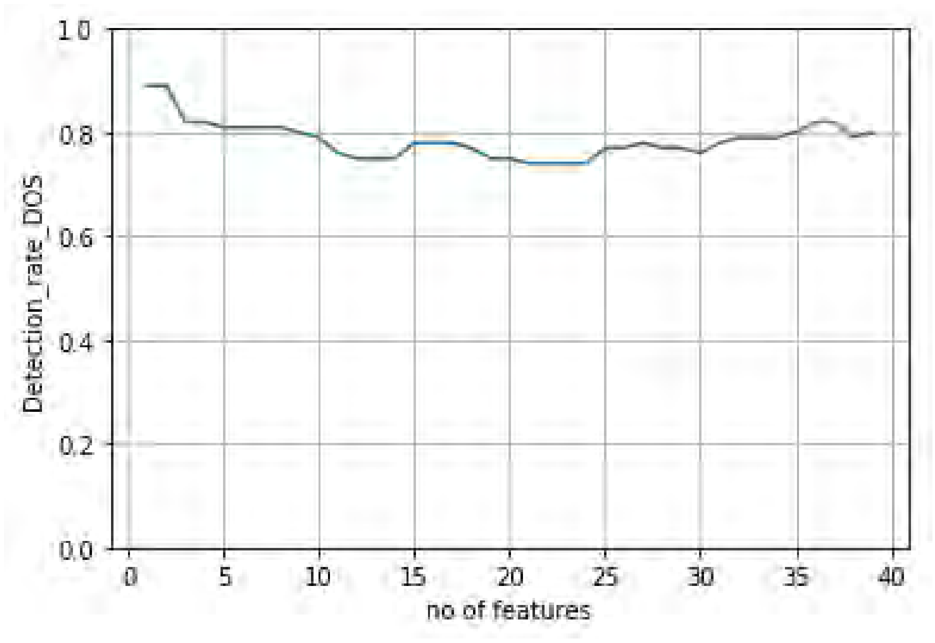
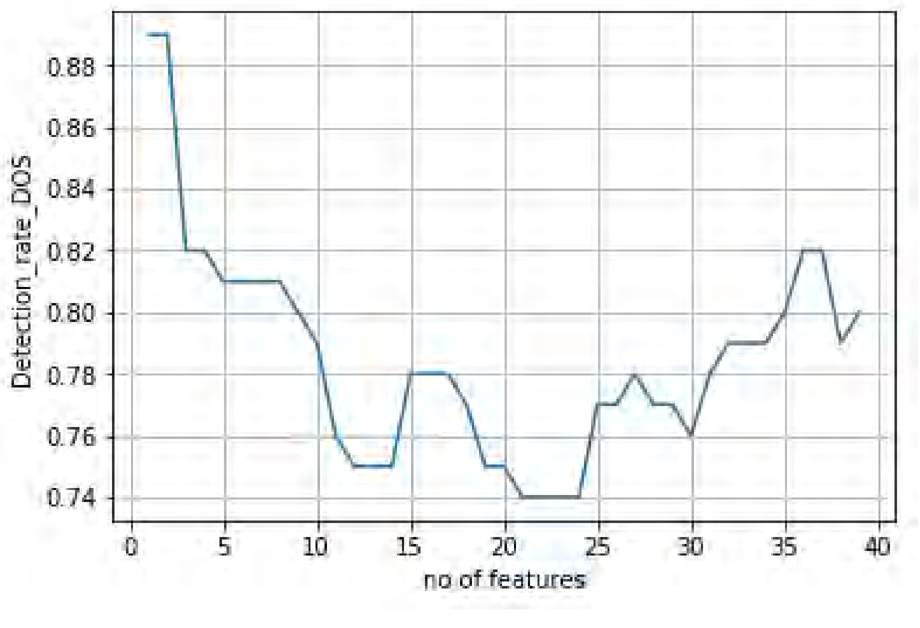


Figure 5.1: DR of Normal traffic by feature selection



(a)



(b)

Figure 5.2: DR of DoS attack by feature selection (a) 0 to 1 scale (b) Zoom view

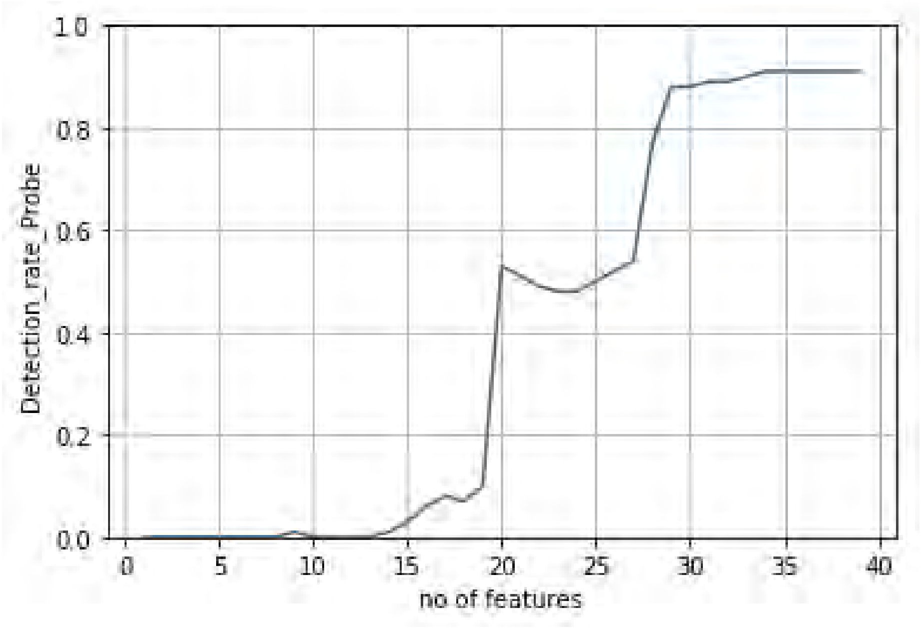


Figure 5.3: DR of Probe attack by feature selection

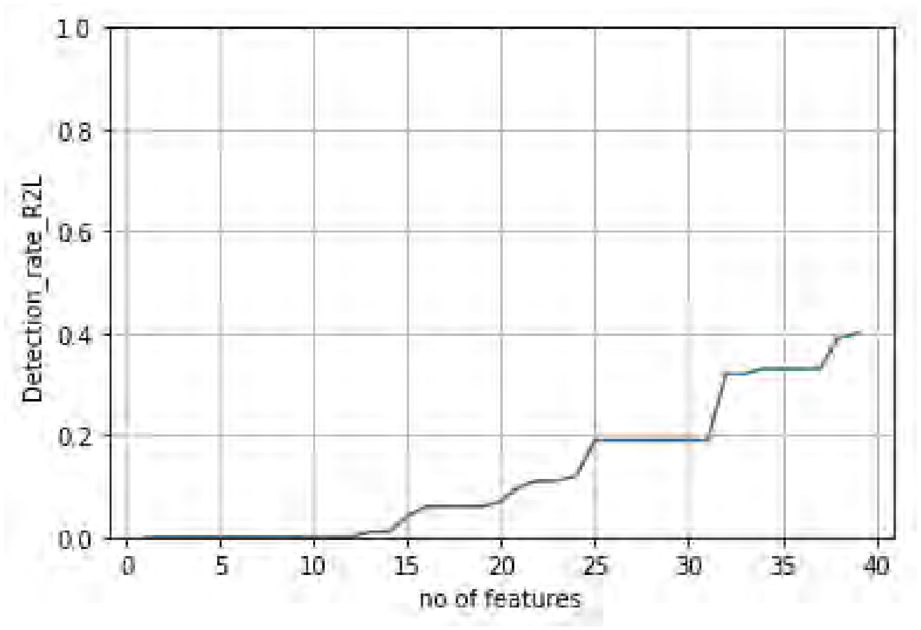


Figure 5.4: DR of R2L attack by feature selection

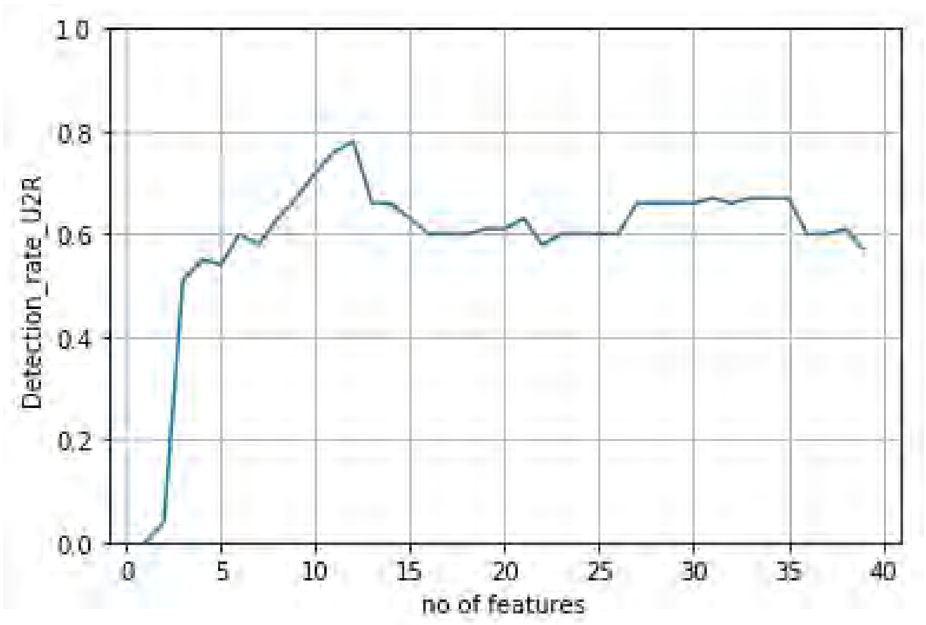
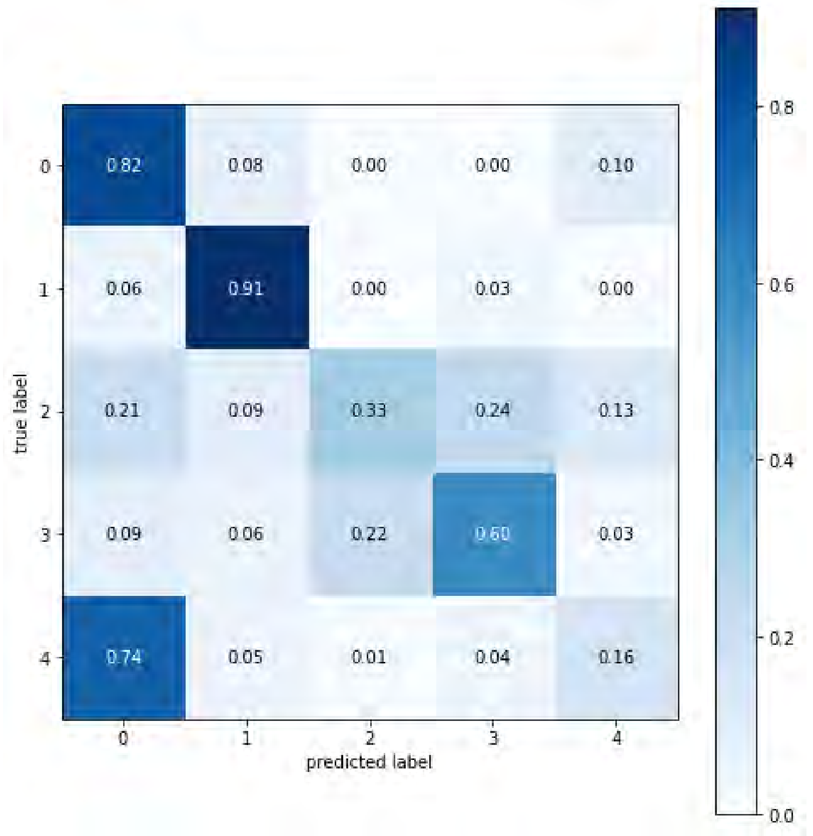


Figure 5.5: DR of U2R attack by feature selection

From Table 5.1, and from Figure 5.1 to Figure 5.5, we can understand that using Bagging_NB classifier, the get the better detection rate using first 36 features. The confusion matrix for the better case is as follows in Figure 5.6.



	0	1	2	3	4
0	0.82	0.08	0.00	0.00	0.10
1	0.06	0.91	0.00	0.03	0.00
2	0.21	0.09	0.33	0.24	0.13
3	0.09	0.06	0.22	0.60	0.03
4	0.74	0.05	0.01	0.04	0.16

Figure 5.6: Confusion matrix of Bagging_NB

Boosting_DT

Boosting_DT is another ensemble classifier. Here, the base classifier of Boosting is Decision Tree. Using this Boosting_DT classifier, the detection rate is represented in Table 5.2. In that case the value of Random State is 1234.

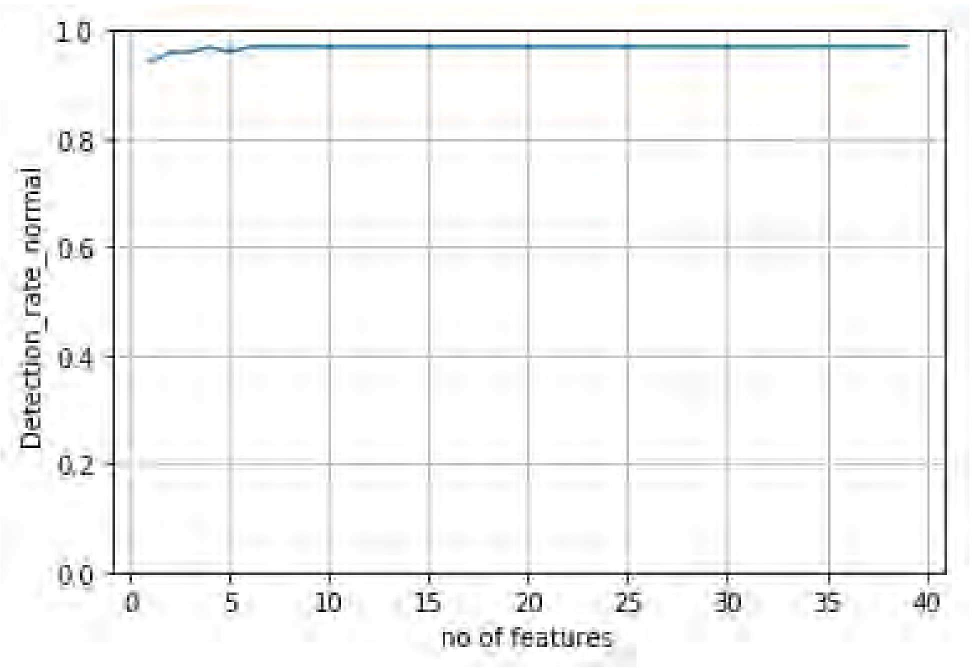
Table 5.2: Detection Rate by feature importance using Boosting_DT classifier

No. of feature	**DoS(0)**	**Probe(1)**	**R2L(2)**	**U2R(3)**	**Normal(4)**
First 1 feature	0.81	0.00	0.00	0.04	0.94
First 2 features	0.76	0.25	0.00	0.15	0.96
First 3 features	0.76	0.23	0.00	0.31	0.96
First 4 features	0.76	0.52	0.03	0.37	0.97
First 5 features	0.72	0.55	0.02	0.28	0.96
First 6 features	0.75	0.55	0.02	0.07	0.97
First 7 features	0.75	0.58	0.06	0.30	0.97
First 8 features	0.75	0.53	0.06	0.15	0.97
First 9 features	0.78	0.61	0.06	0.09	0.97
First 10 features	0.77	0.55	0.06	0.12	0.97
First 11 features	0.76	0.55	0.06	0.12	0.97
First 12 features	0.77	0.58	0.05	0.10	0.97
First 13 features	0.80	0.62	0.07	0.10	0.97
First 14 features	0.76	0.60	0.01	0.06	0.97
First 15 features	0.77	0.62	0.00	0.07	0.97

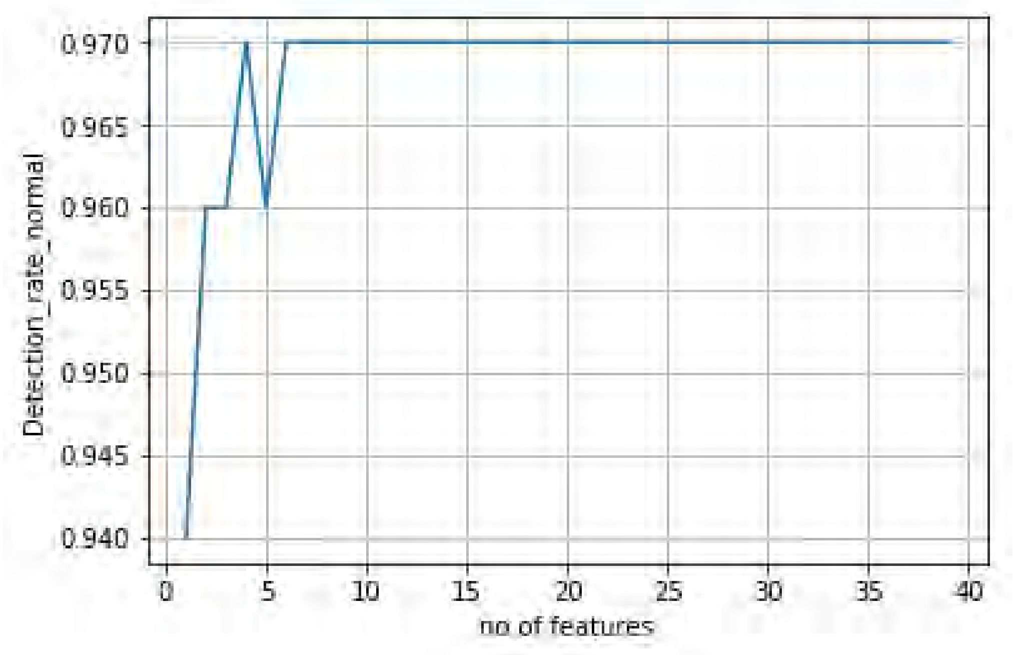
No. of feature	DoS(0)	Probe(1)	R2L(2)	U2R(3)	Normal(4)
First 16 features	0.76	0.60	0.01	0.07	0.97
First 17 features	0.78	0.60	0.01	0.04	0.97
First 18 features	0.77	0.61	0.01	0.06	0.97
First 19 features	0.81	0.57	0.01	0.03	0.97
First 20 features	0.74	0.59	0.03	0.03	0.97
First 21 features	0.77	0.59	0.01	0.04	0.97
First 22 features	0.77	0.59	0.04	0.03	0.97
First 23 features	0.75	0.60	0.04	0.04	0.97
First 24 features	0.75	0.61	0.03	0.03	0.97
First 25 features	0.75	0.59	0.01	0.03	0.97
First 26 features	0.75	0.60	0.01	0.03	0.97
First 27 features	0.75	0.62	0.01	0.06	0.97
First 28 features	0.75	0.62	0.01	0.04	0.97
First 29 features	0.75	0.60	0.03	0.04	0.97
First 30 features	0.75	0.60	0.02	0.06	0.97
First 31 features	0.77	0.59	0.02	0.04	0.97
First 32 features	0.75	0.62	0.01	0.04	0.97
First 33 features	0.80	0.59	0.02	0.06	0.97

No. of feature	DoS(0)	Probe(1)	R2L(2)	U2R(3)	Normal(4)
First 34 features	0.76	0.60	0.02	0.07	0.97
First 35 features	0.78	0.61	0.03	0.12	0.97
First 36 features	0.78	0.60	0.01	0.06	0.97
First 37 features	0.78	0.65	0.02	0.07	0.97
First 38 features	0.82	0.63	0.05	0.03	0.97
First 39 features	0.81	0.62	0.03	0.09	0.97

The Table 5.2 is graphically represented from Figure 5.7 to Figure 5.11.

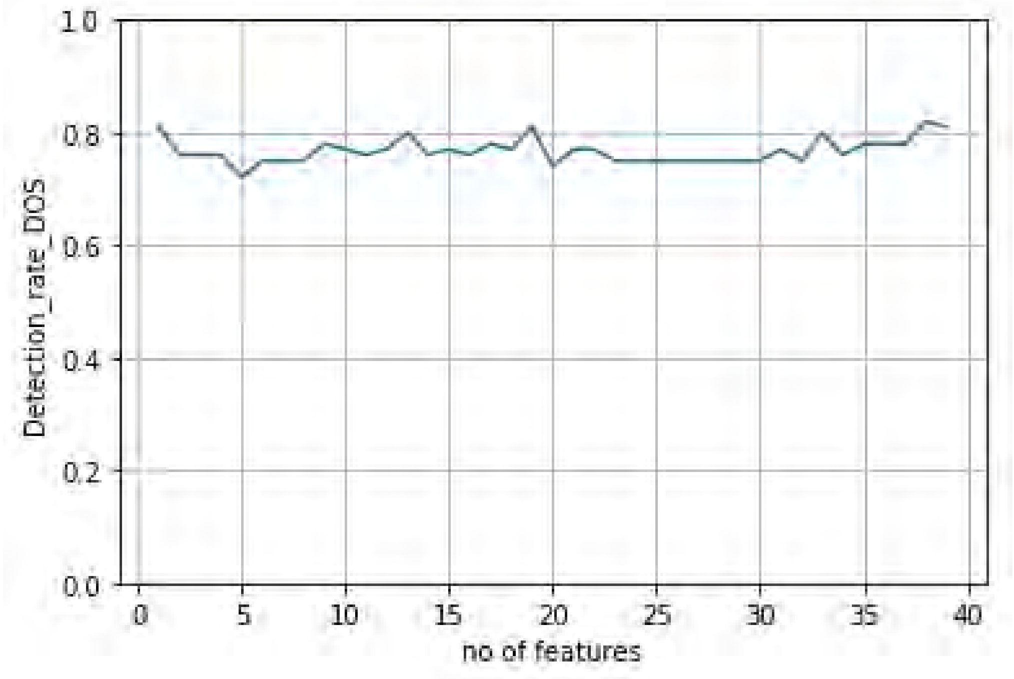


(a)

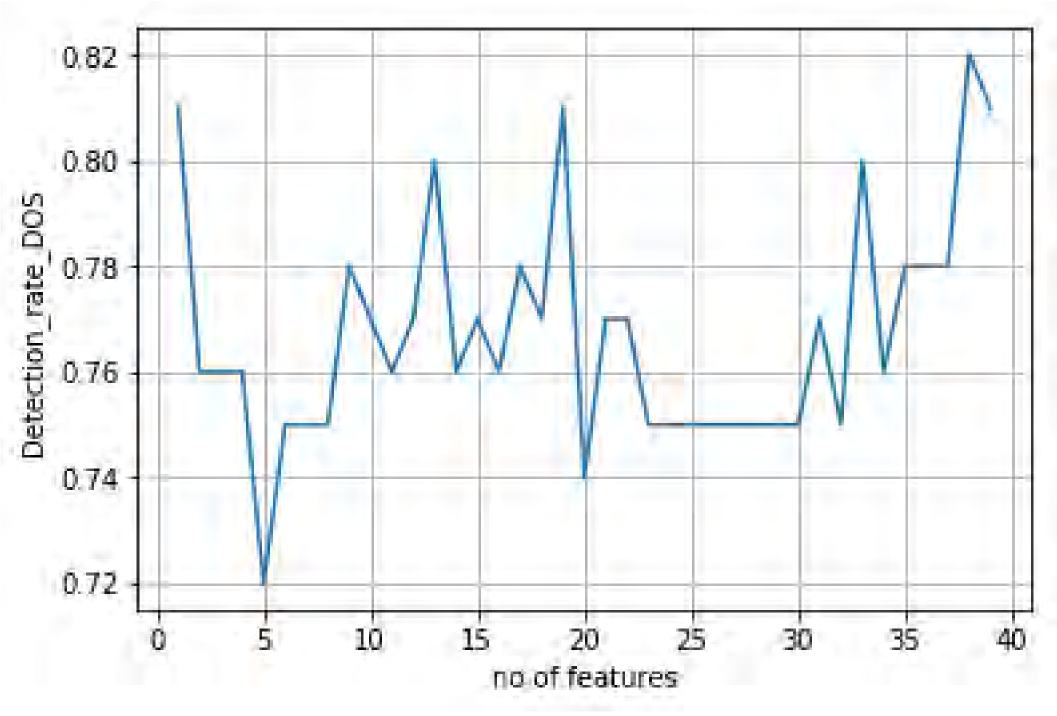


(b)

Figure 5.7: DR of Normal Traffic by feature selection (a) 0 to 1 scale (b) Zoom view

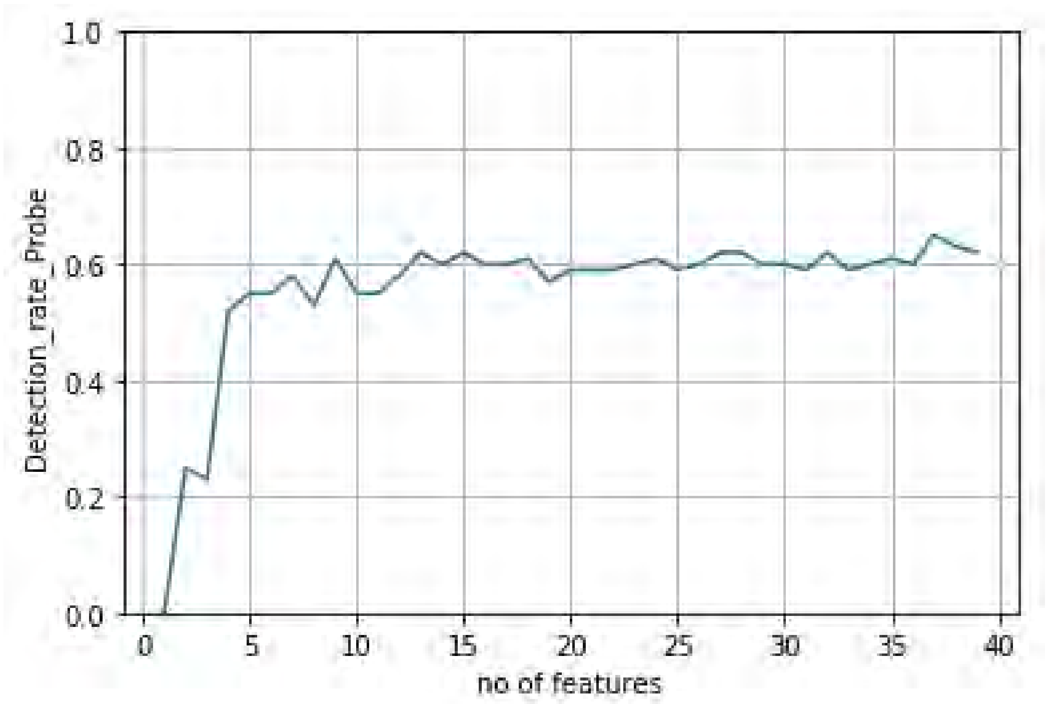


(a)

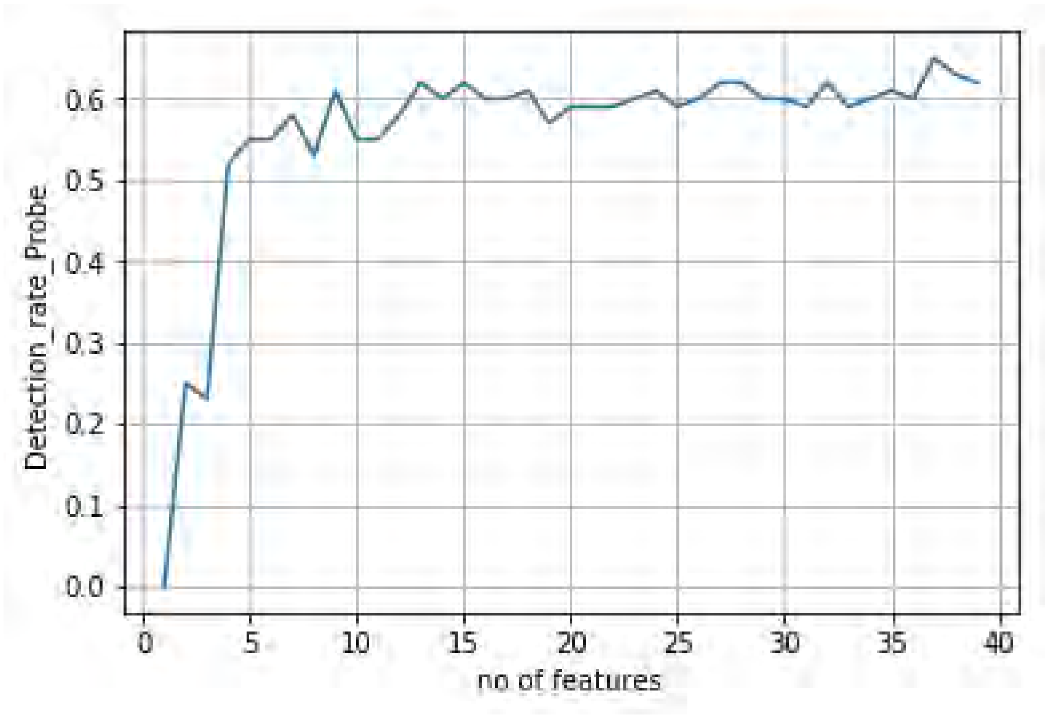


(b)

Figure 5.8: DR of DoS attack by feature selection (a) 0 to 1 scale (b) Zoom view

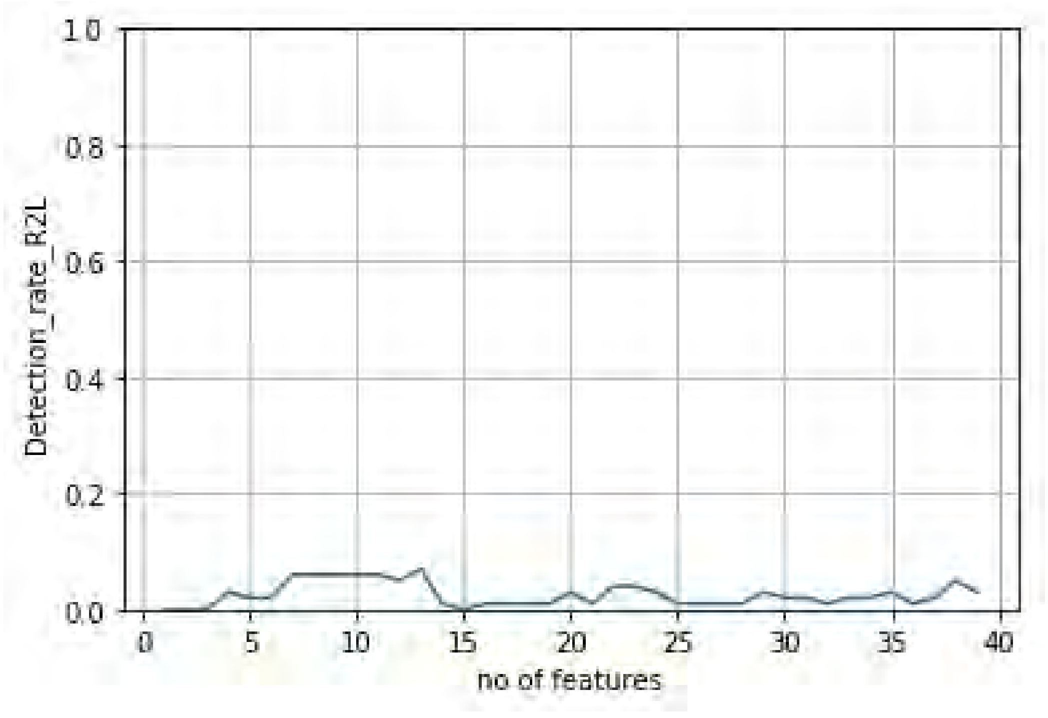


(a)

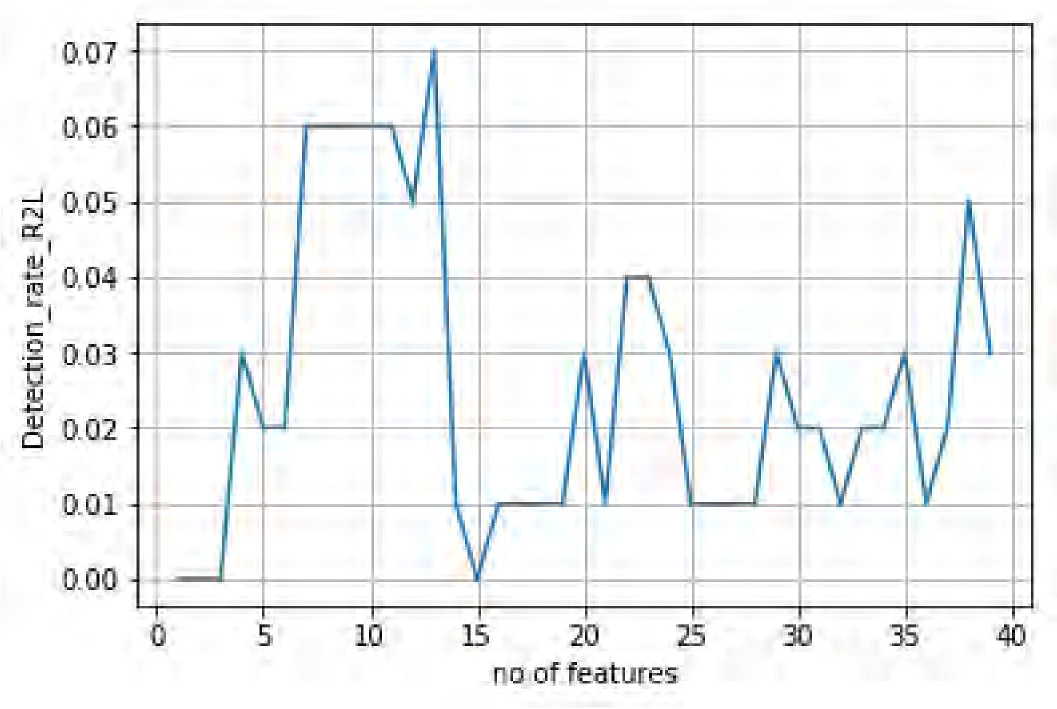


(b)

Figure 5.9: DR of Probe attack by feature selection (a) 0 to 1 scale (b) Zoom view

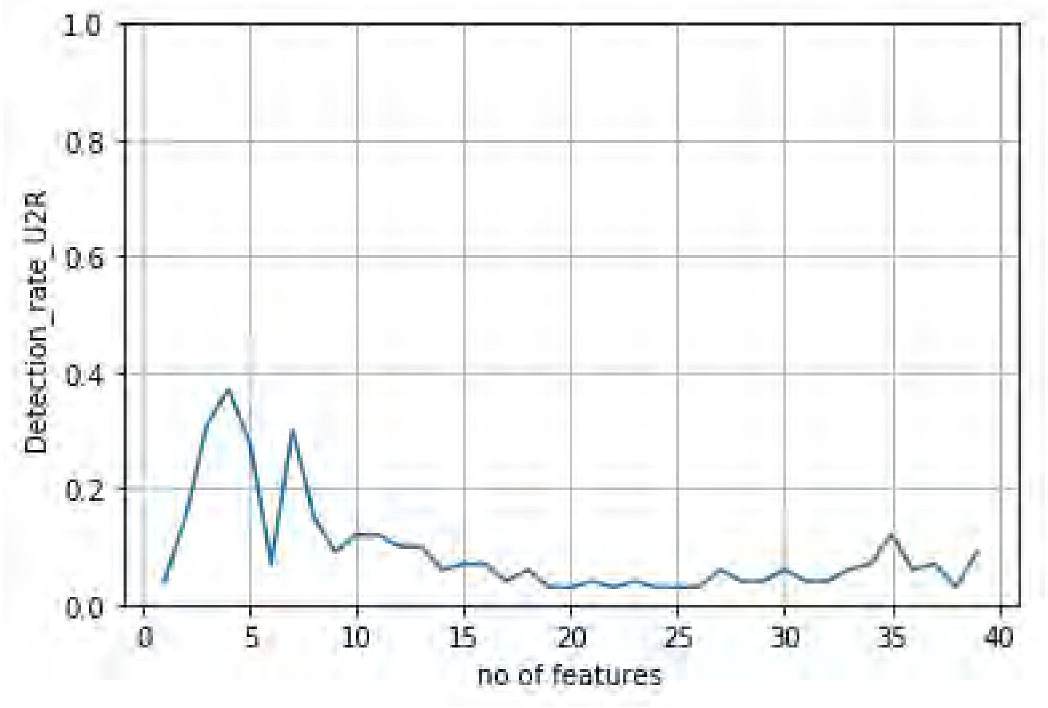


(a)

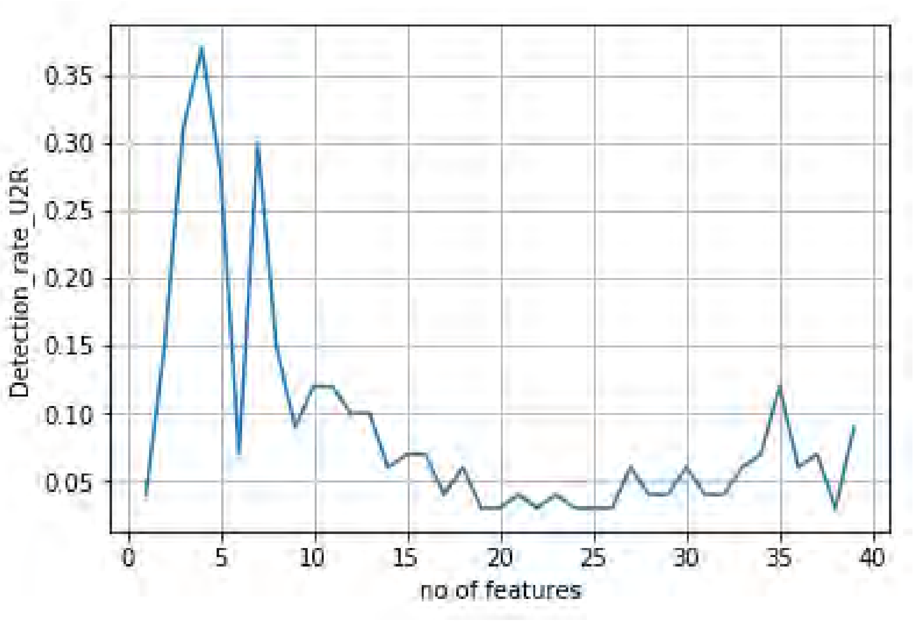


(b)

Figure 5.10: DR of R2L attack by feature selection (a) 0 to 1 scale (b) Zoom view



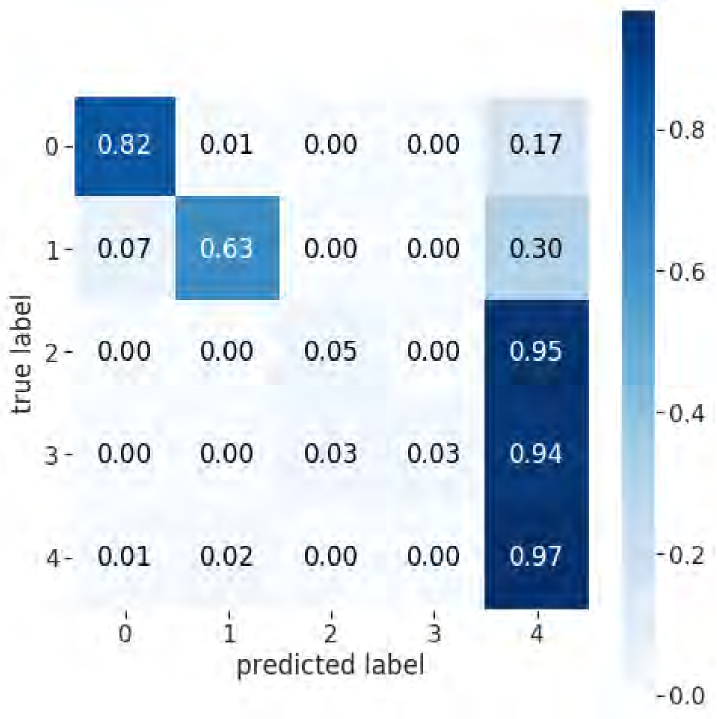
(a)



(b)

Figure 5.11: DR of U2R attack by feature selection (a) 0 to 1 scale (b) Zoom view

From Figure 5.6 to Figure 5.11, we can understand that the combination of first 38 features get the better detection rate that other features combination when used the Boosting_DT ensemble classifier. The confusion matrix for the better case is as follows in Figure 5.12.



true label \ predicted label	0	1	2	3	4
0	0.82	0.01	0.00	0.00	0.17
1	0.07	0.63	0.00	0.00	0.30
2	0.00	0.00	0.05	0.00	0.95
3	0.00	0.00	0.03	0.03	0.94
4	0.01	0.02	0.00	0.00	0.97

Figure 5.12: Confusion Matrix of Boosting_DT

Voting

The base classes of voting classifiers are DT, KNN and LR. The detection rate of voting classifier using the first 34 features is presented in Table 5.3. The confusion matrix of this classifier is presented below in Figure 5.13.

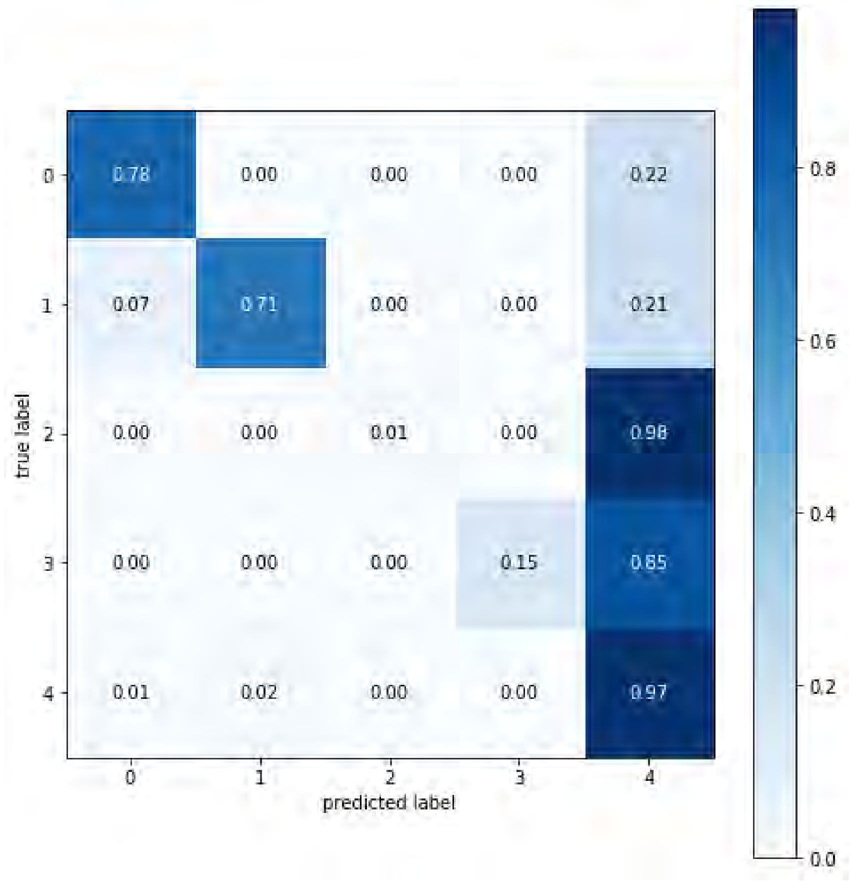


Figure 5.13: Confusion Matrix of Voting classifier

We compare with only recent literature that implement ensemble machine learning classifiers and also measure the detection rate. Table 5.3 shows our implemented result and the literature result of detection rate. From Table 5.3, we can see that in 2019 the Random Forest ensemble machine learning classifier implemented to find out the detection rate. In that table, we show three ensemble machine learning classifiers namely Boosting_DT, Voting and Bagging_NB. For Boosting_DT, the base class of Boosting is Decision Tree; for Voting, the base classes are same as Voting-1 (Figure 4.3); for Bagging_NB, the base class of Bagging classifier is Naïve Bayes.

Table 5.3: DR of ensemble methods without cross validation

Classifiers	Normal (%)	DoS (%)	Probe (%)	R2L (%)	U2R (%)
Random Forest [111]	97.37	81.47	68.86	2.73	1.50
Boosting_DT	97	82	63	5	3
Voting	97	78	71	01	15
Bagging_NB	16	82	91	33	60

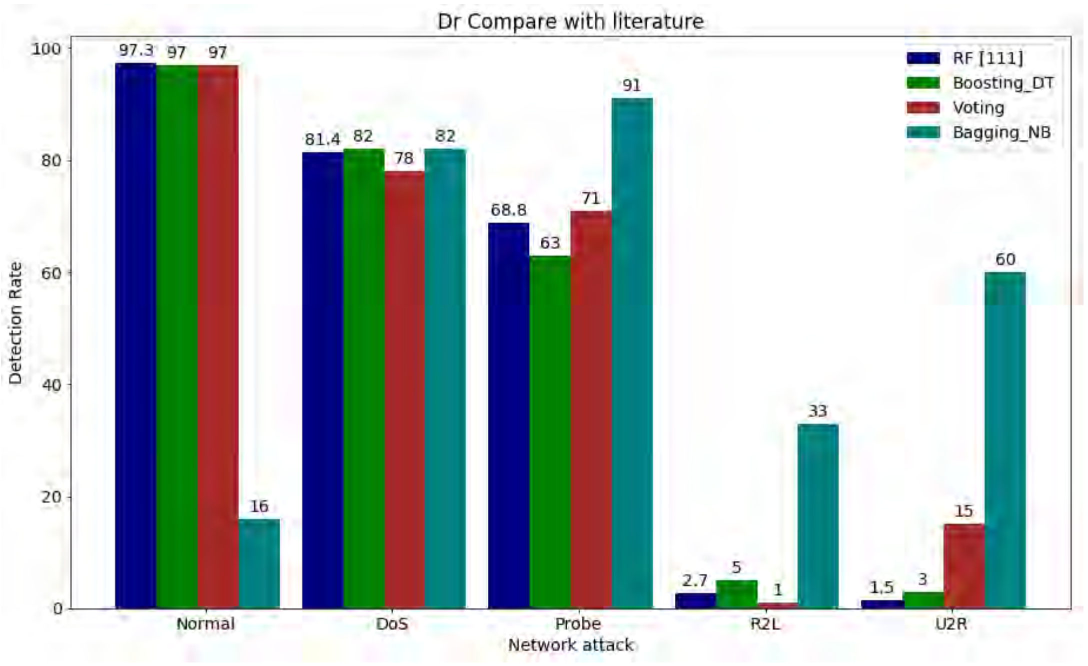


Figure 5.14: DR of Voting methods without cross validation

From Figure 5.14, it is clear that among four types of ensemble classifiers, the Bagging_NB performs the best in case to detect the DoS, Probe, R2l and U2R attacks; however, the detection rate of normal traffic is low of Bagging_NB classifiers. Thus, Boosting_DT performs also similar in the case to detect the normal traffic and DoS attack. This classifier performs better than recent literature to detect R2L and U2R, the only exception is to detect the probe attack. In that case, we would like to propose the Boosting_DT. To execute this work, we have used the full training and testing dataset separately.

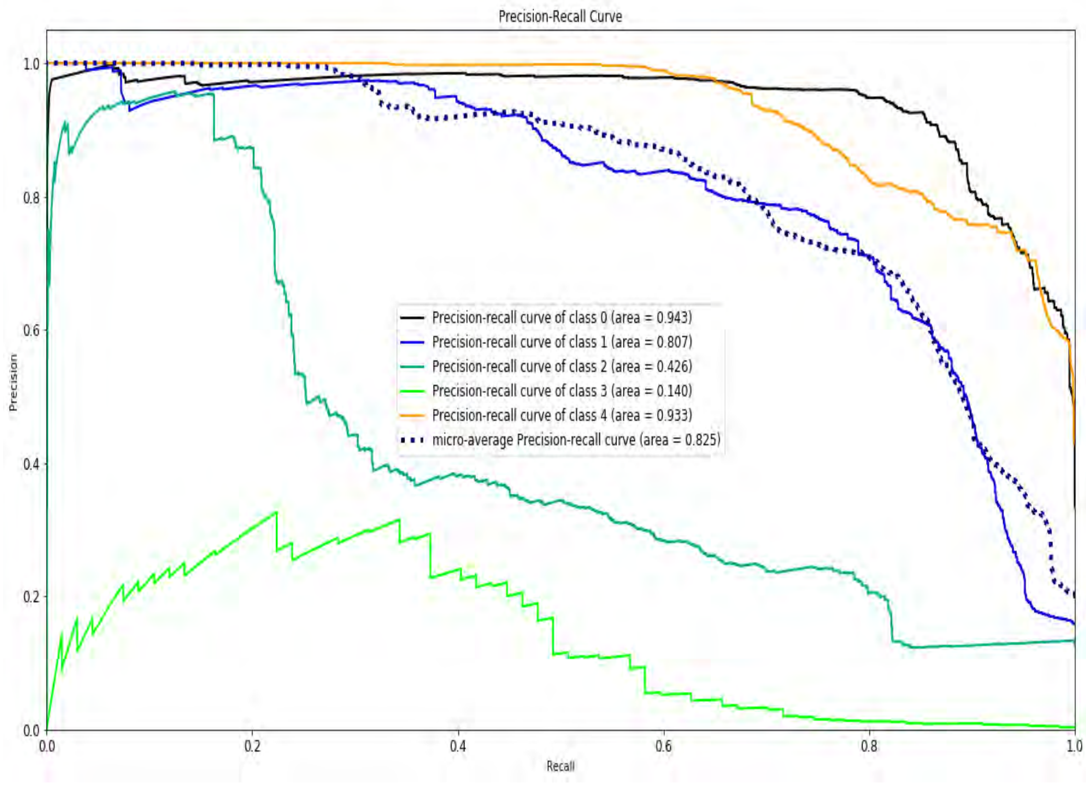


Figure 5.15: Precision-Recall Curve of Boosting_DT

From the Precision-Recall curve (Figure 5.15), we can observe the individual area of detection of normal traffic and 4 types of attacks. Here, the area of R2L attack (class 3, Figure 5.15) was the lowest and area for DoS attack (Class 0, Figure 5.15) was the highest.

5.2.2 Detection Using Cross Validation

In this step, 3-fold cross validation has been implemented. At first, using 3-fold cross validation, the Voting-1 ensemble classifier is implemented. For Voting-1, the base classifier is the combination of Decision Tree (DT), K Nearest Neighbour (KNN) and Logistic Regression (LR). According to the importance of features (p-value, described in chapter 5) every time one more feature has been added and Voting-1 ensemble classifier is executed.

5.2.2.1 Voting

Voting methods discussed in section 3.6.3 and our proposed design depicted in section 3.7.1. Four types of voting and their base classes are shown in Table 5.4.

Table 5.4: DR of voting methods without cross validation

Methodology	Base Classifier-1	Base Classifier-2	Base Classifier-3	Voting type
Voting-1	DT	KNN	LR	Hard
Voting-2	DT	SVC(kernel = Poly)	LR	Hard
Voting-3	DT	SVC(kernel = RBF)	LR	Hard
Voting-4	DT	SVC(kernel = Linear)	LR	Hard

5.2.2.1.1 Findings of Voting-1

For voting-1 method, the findings in terms of detection rate has shown in Table 5.5

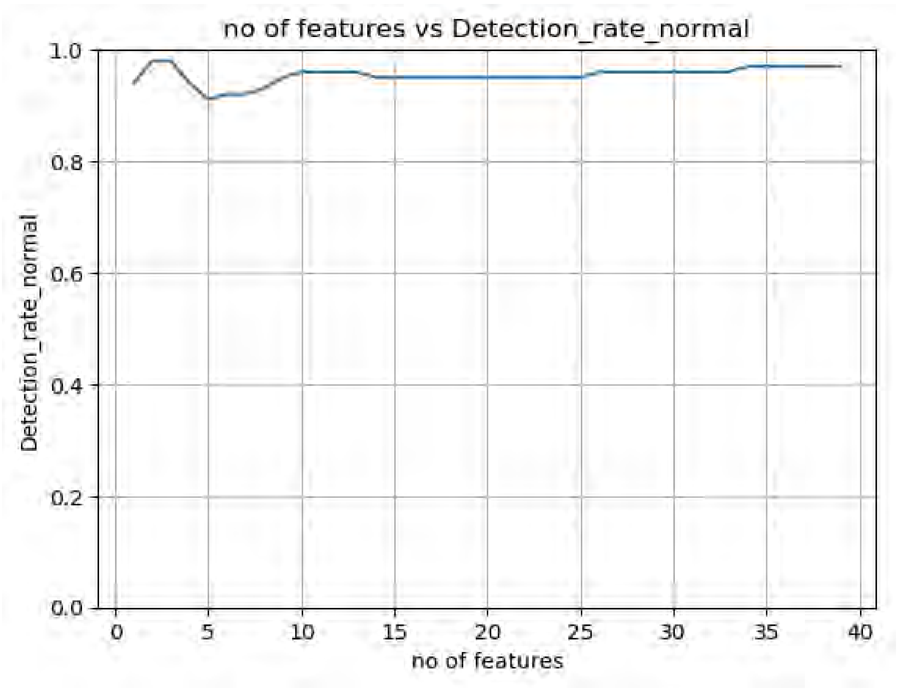
Table 5.5: Detection Rate by feature importance

No. of feature	DoS(0)	Probe(1)	R2L(2)	U2R(3)	Normal(4)
First 1 feature	0.99	0.22	0.79	0.06	0.94
First 2 features	0.99	0.32	0.89	0.10	0.98
First 3 features	1.00	0.38	0.88	0.07	0.98
First 4 features	0.97	0.83	0.88	0.12	0.94
First 5 features	0.93	0.83	0.82	0.04	0.91
First 6 features	0.95	0.86	0.83	0.04	0.92
First 7 features	0.95	0.86	0.81	0.00	0.92
First 8 features	0.95	0.83	0.87	0.00	0.93
First 9 features	0.96	0.90	0.89	0.00	0.95
First 10 features	0.96	0.90	0.89	0.00	0.96

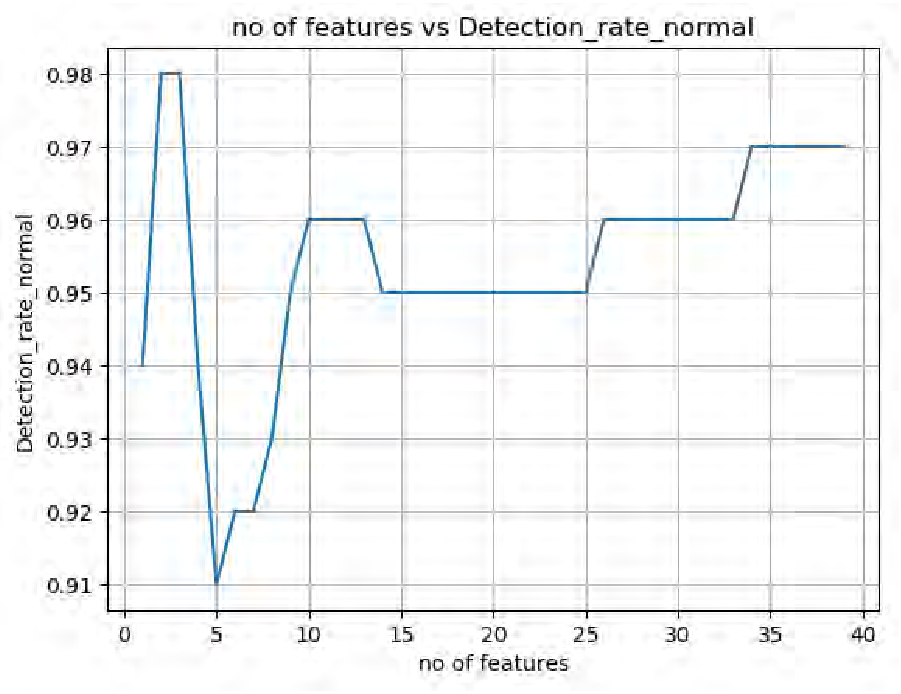
No. of feature	DoS(0)	Probe(1)	R2L(2)	U2R(3)	Normal(4)
First 11 features	0.96	0.91	0.89	0.01	0.96
First 12 features	0.96	0.92	0.89	0.01	0.96
First 13 features	0.96	0.92	0.89	0.09	0.96
First 14 features	0.97	0.94	0.92	0.12	0.95
First 15 features	0.97	0.95	0.92	0.16	0.95
First 16 features	0.97	0.95	0.92	0.19	0.95
First 17 features	0.97	0.94	0.92	0.15	0.95
First 18 features	0.97	0.95	0.93	0.18	0.95
First 19 features	0.99	0.95	0.93	0.13	0.95
First 20 features	0.99	0.96	0.94	0.12	0.95
First 21 features	0.99	0.96	0.94	0.10	0.95
First 22 features	0.99	0.96	0.94	0.09	0.95
First 23 features	0.99	0.96	0.94	0.10	0.95
First 24 features	0.99	0.97	0.95	0.12	0.95
First 25 features	0.99	0.96	0.94	0.25	0.95
First 26 features	0.99	0.97	0.94	0.27	0.96
First 27 features	0.99	0.97	0.94	0.40	0.96
First 28 features	0.99	0.97	0.94	0.42	0.96

No. of feature	DoS(0)	Probe(1)	R2L(2)	U2R(3)	Normal(4)
First 29 features	0.99	0.97	0.94	0.40	0.96
First 30 features	0.99	0.97	0.95	0.40	0.96
First 31 features	0.99	0.97	0.95	0.49	0.96
First 32 features	0.99	0.97	0.94	0.48	0.96
First 33 features	0.99	0.97	0.94	0.55	0.96
First 34 features	0.99	0.97	0.94	0.55	0.97
First 35 features	1.00	0.97	0.94	0.54	0.97
First 36 features	1.00	0.97	0.94	0.57	0.97
First 37 features	0.99	0.97	0.94	0.55	0.97
First 38 features	1.00	0.97	0.94	0.55	0.97
First 39 features	1.00	0.97	0.94	0.57	0.97

It is easy to understand Table 5.5 by graphical visualization from Figure 5.16 to Figure 5.20.



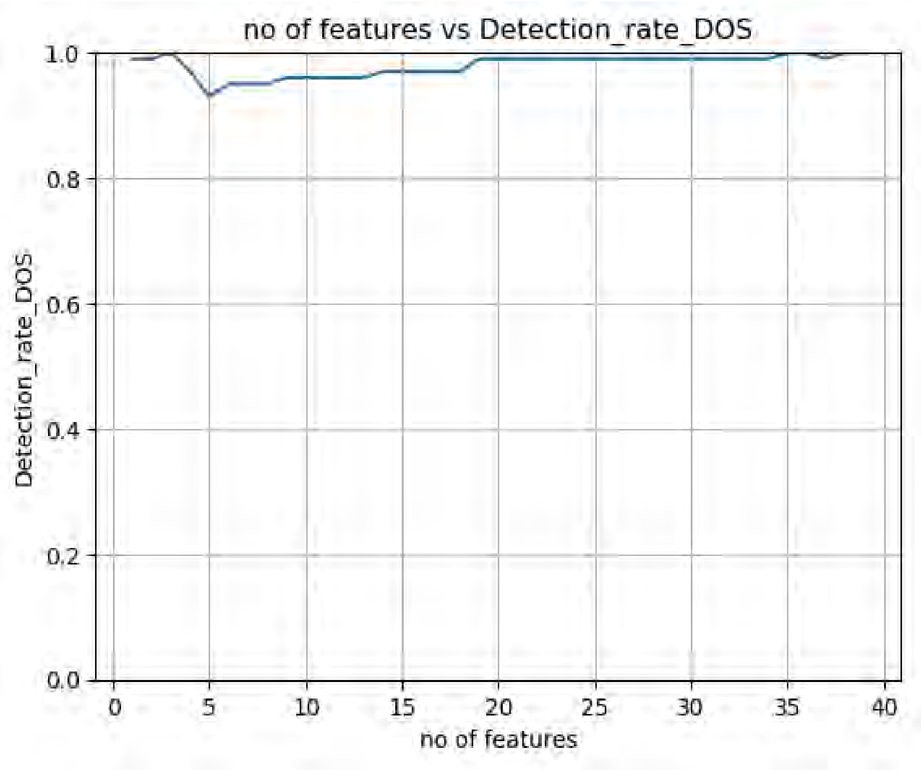
(a)



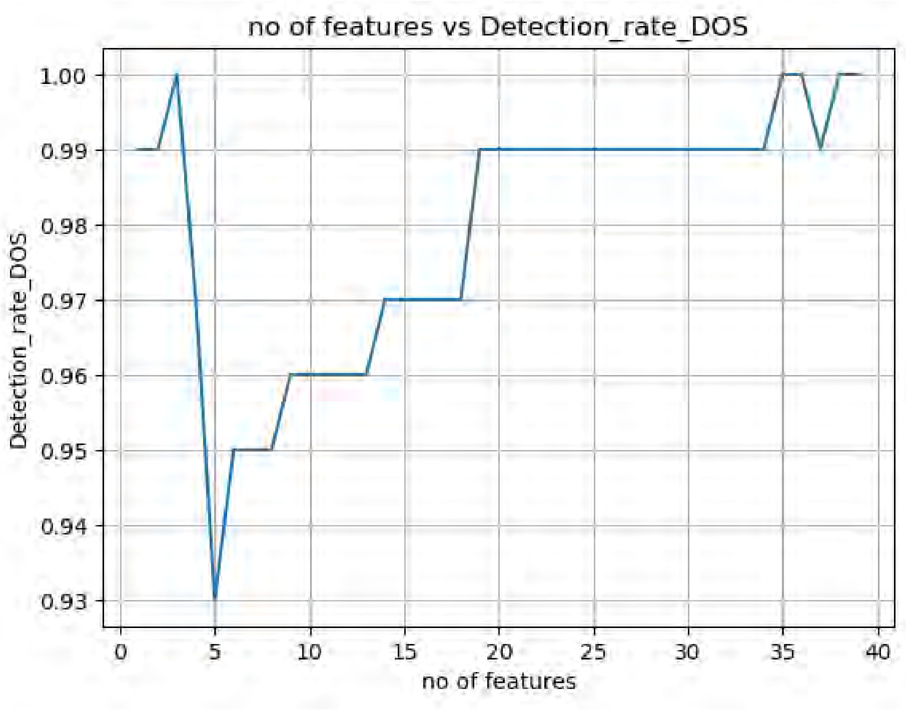
(b)

Figure 5.16: DR of Normal traffic by feature selection (a) 0 to 1 scale (b) Zoom view

It is easy to understand from Table 5.5 and Figure 5.16 that only first 2 or 3 features are enough to detect the normal traffic with 98% detection accuracy using voting-1 algorithm. Using 36 features, we again get the better detection rate. The lowest detection rate is obtained when number of features are 5 which indicate that the number 5 feature contains some noise data to predict the normal traffic. The number 5 features make the classifier get confused to predict the normal traffic accurately. We will observe the same scenario in Figure 5.17 and Figure 5.19. Now, we will show how DR rate is changing with the number of features to detect the DoS, Probe, R2L and U2R attacks.



(a)



(b)

Figure 5.17: DR of DoS attack by feature selection (a) 0 to 1 scale (b) Zoom view

From Figure 5.17 and Table 5.5, it is clear that only first 3 features are enough to detect DoS attack with 100% detection accuracy using voting-1 algorithm.

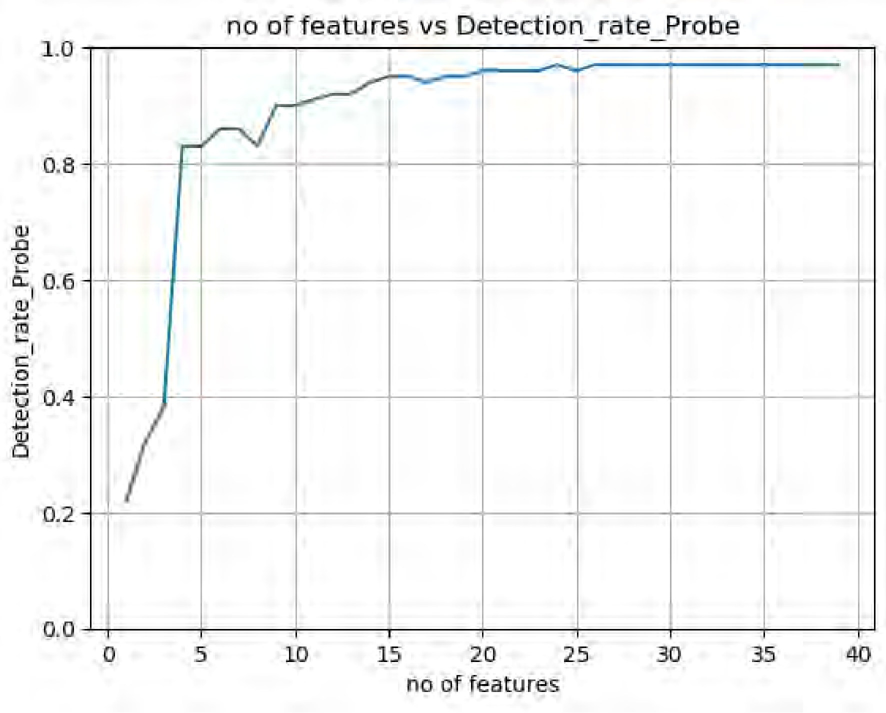
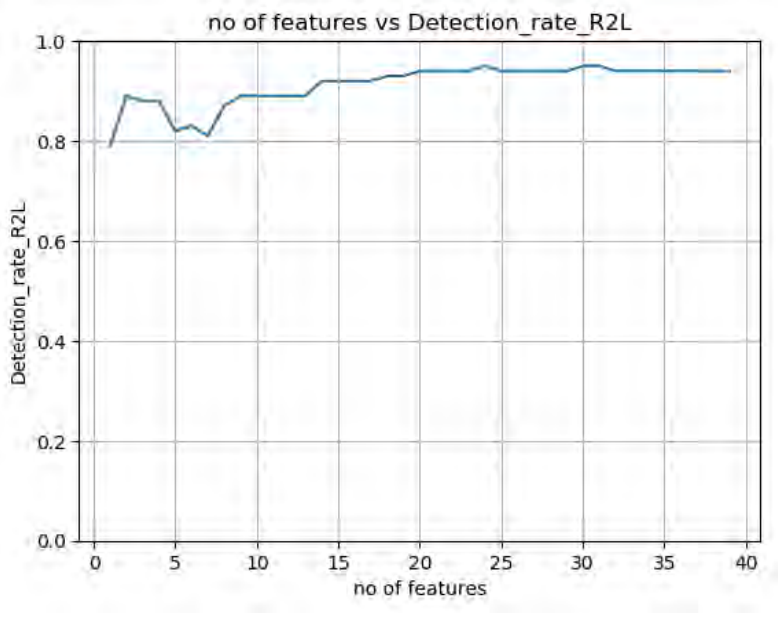
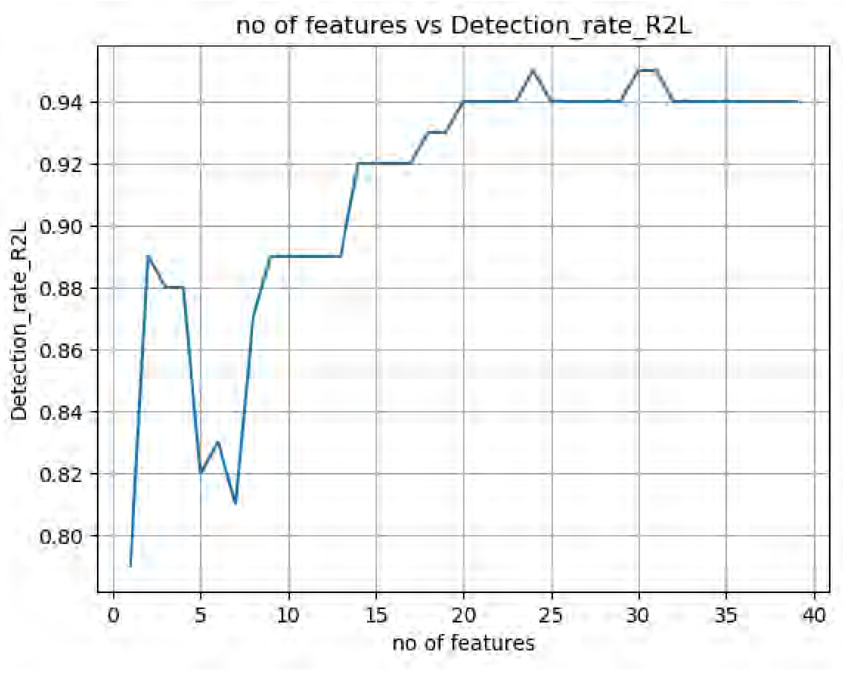


Figure 5.18: DR of Probe attack by feature selection

From Figure 5.18, it can be clearly seen that using first 5 important features the detection rate of probe attack increases rapidly. Using first 5 to 25 important features the detection rate increases slowly. After taking first 26 important features the detection rate is constant. So, first 26 important features are enough to get the highest detection rate of probe attack using voting-1 algorithm.



(a)



(b)

Figure 5.19: DR of R2L attack by feature selection (a) 0 to 1 scale (b) zoom view

From Figure 5.19, it is clear that Voting-1 provide the highest detection rate by 95% when it takes first 24 important features and also when takes first 30 important features.

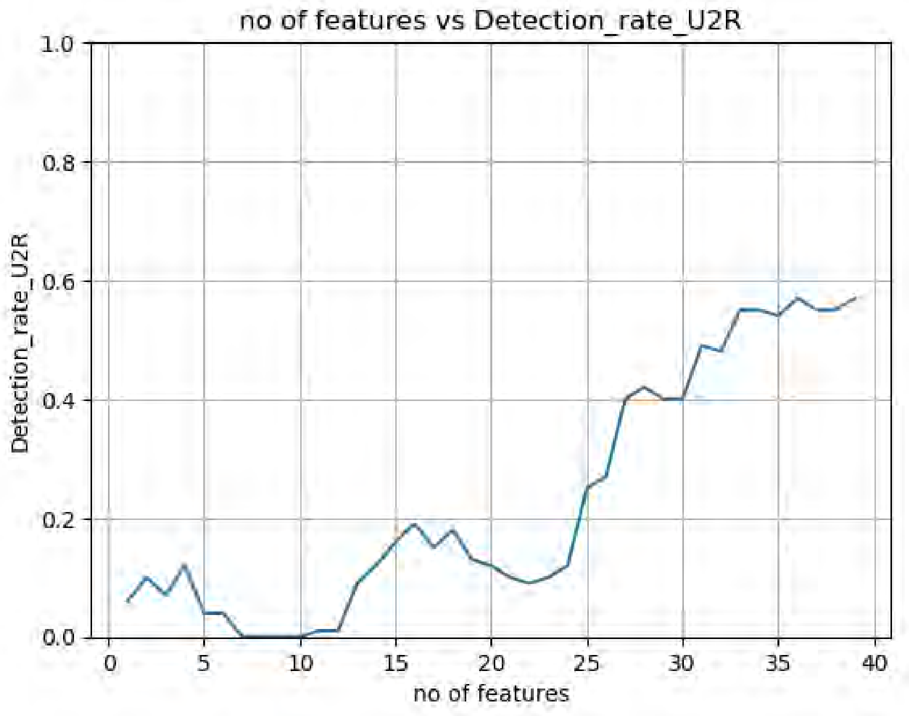


Figure 5.20: DR of U2R attack by feature selection

From Figure 5.20, it can be seen that using lower number of features, voting-1 does not perform well to detect U2R attack. The highest detection rate is 57% when it takes first 36 important features.

Form Figure 5.16 to 5.20, we can see that using first 36 important features can perform better if it is trained by the classifier to detect normal traffic and four types of network attacks.

After implementing voting-1 classifier, another three voting classifiers has been implemented. For all of the voting classifiers, their base classifiers is shown in Table 5.4

5.2.2.1.2 Performance of Voting Methods

Confusion matrix of four voting classifiers is shown in Figure 5.21, Figure 5.22, Figure 5.23 and Figure 5.24.

Confusion Matrix of Voting-1

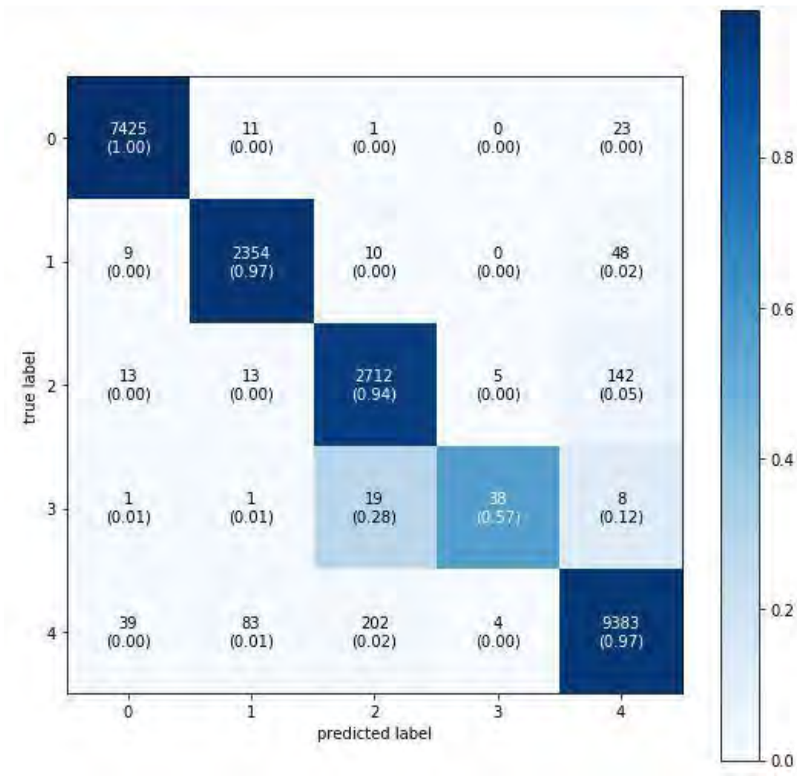


Figure 5.21: Confusion matrix of Voting-1 (using 36 features)

Confusion Matrix of Voting-2

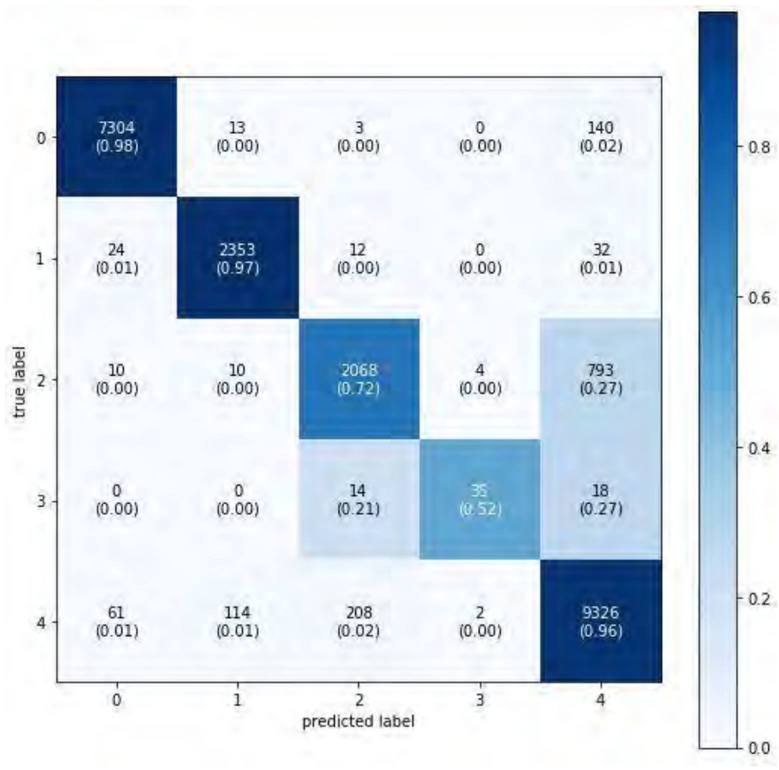


Figure 5.22: Confusion matrix of Voting - 2 (using 36 features)

Confusion Matrix of Voting-3:

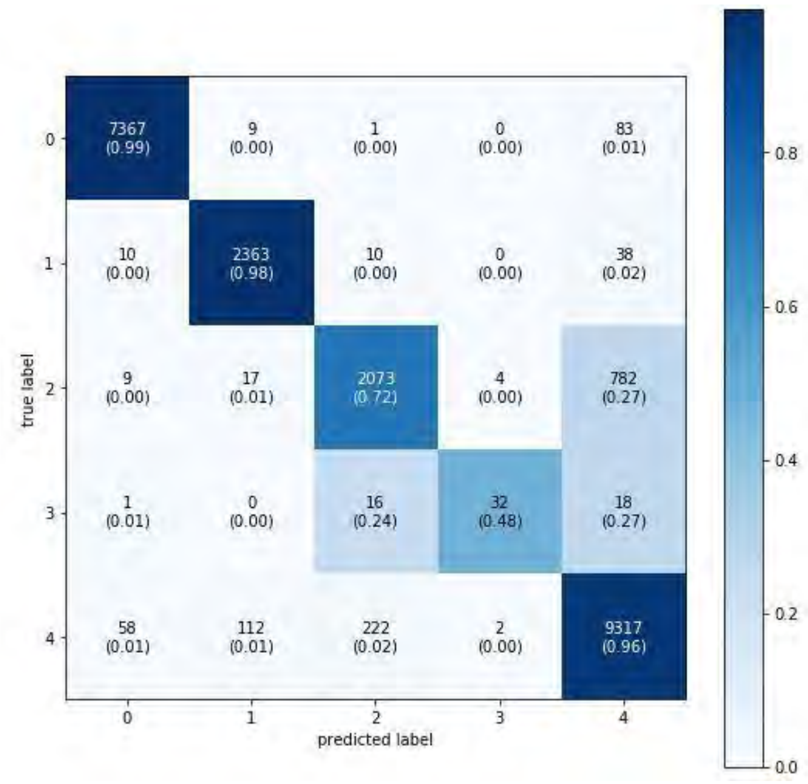


Figure 5.23: Confusion matrix of Voting - 3 (using 36 features)

Confusion Matrix of Voting-4

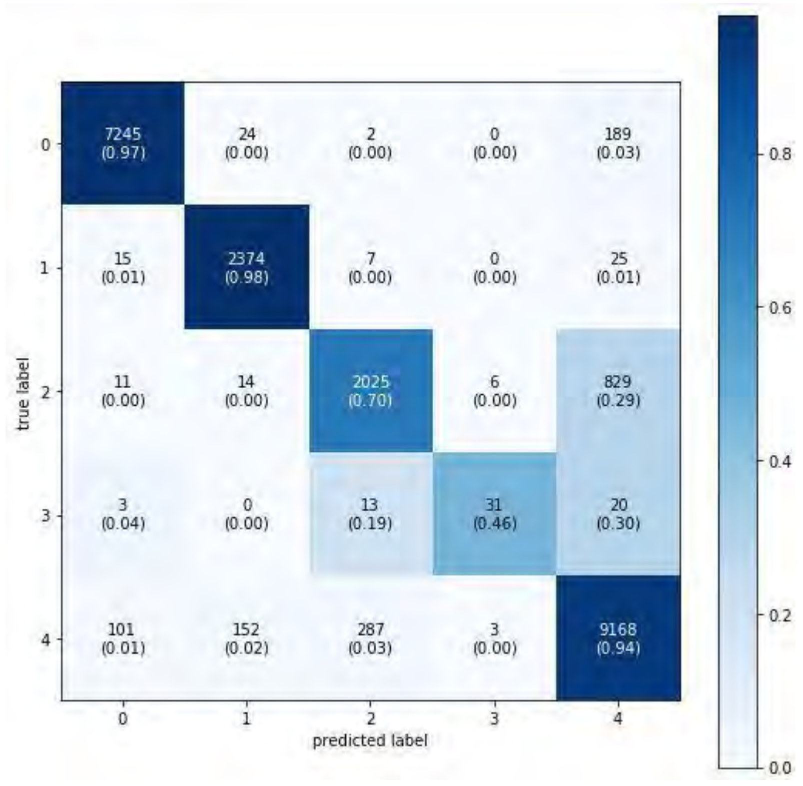


Figure 5.24: Confusion matrix of Voting -4 (using 36 features)

5.2.2.1.3 Detection Rate of Voting Methods

Detection rate of normal traffic and four types of attacks using four types of voting classifiers is shown in Table 5.6.

Table 5.6: Detection rate for Voting with 36 features

Classifier	Normal (4)	DoS (0)	Probe (1)	R2L (2)	U2R (3)
Voting-1	0.97	1.00	0.97	0.94	0.57
Voting-2	0.96	0.98	0.97	0.72	0.52
Voting-3	0.96	0.99	0.98	0.72	0.48
Voting-4	0.94	0.97	0.98	0.70	0.46

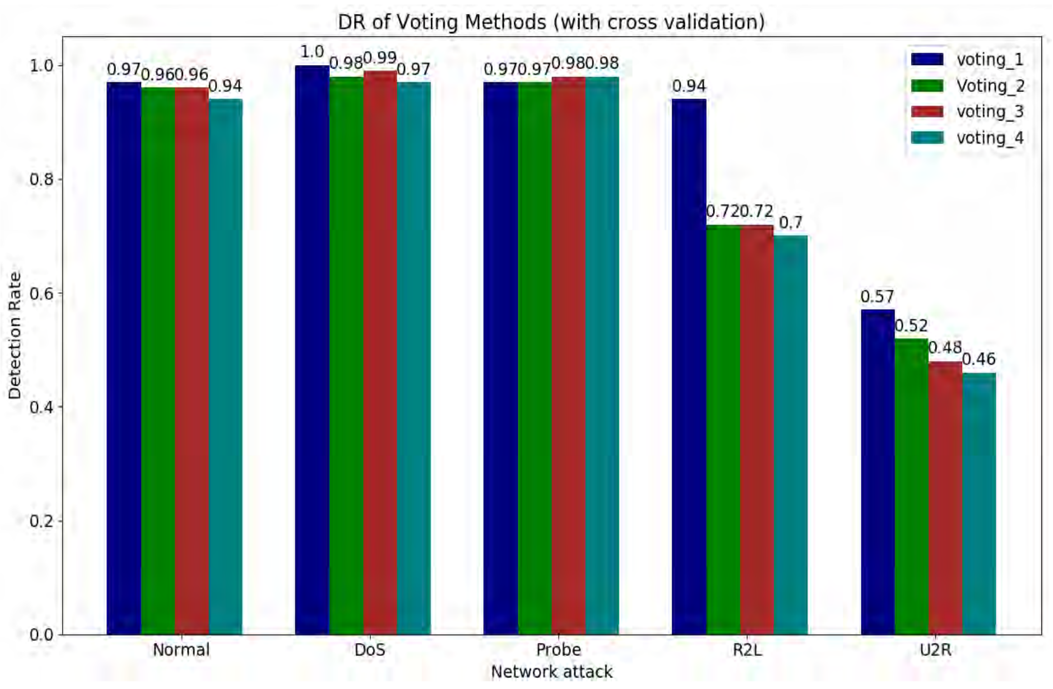


Figure 5.25: Detection rate of Voting classifiers with 36 features

Using cross validation, most of the cases voting-1 perform well except probe attack detection. Interestingly, this classifier performs very well in the case of R2L and U2R attack detection.

5.2.2.2 Stacking

Stacking method discussed in section 3.6.4 and our proposed design depicted in section 3.7.2. As mentioned earlier that the stacking classifier can take more than one classifier and take a meta classifier to predict from the base classifiers. Table 5.7 shows four types stacking classifier arrangements

Table 5.7: Four Stacking arrangements

Classifier	Base Classifier-1	Base Classifier-2	Meta classifier
Stacking-1	SVC (kernel = Linear)	LR	SVC (kernel = Linear)
Stacking-2	SVC (kernel = RBF)	LR	SVC (kernel = RBF)
Stacking-3	SVC (kernel = RBF)	LR	NB
Stacking-4	SVC (kernel = RBF)	LR	RandomForest

Confusion matrix of four types of stacking classifiers is shown in Figure 5.26, Figure 5.27, Figure 5.28 and Figure 5.29.

Confusion Matrix of Stacking-1

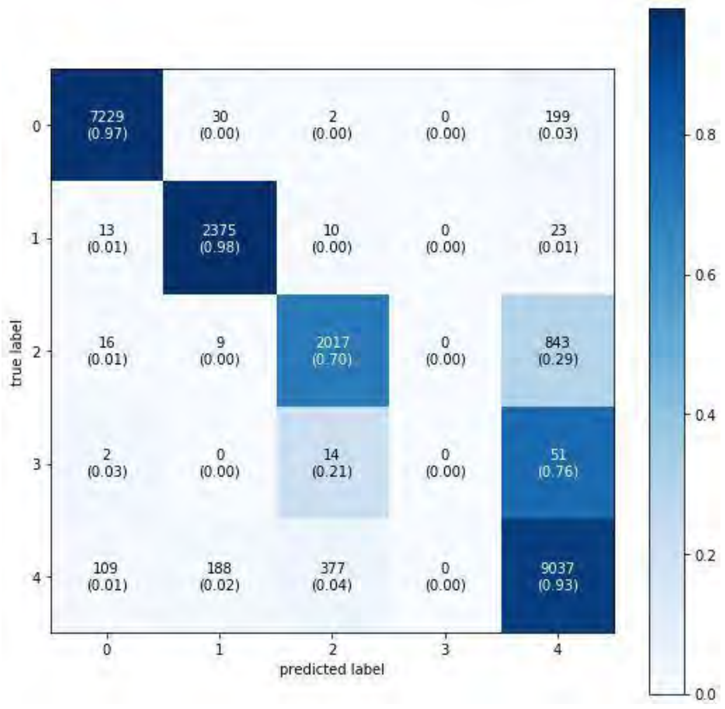


Figure 5.26: Confusion matrix of Stacking-1 (using 36 features)

Confusion Matrix of Stacking-2

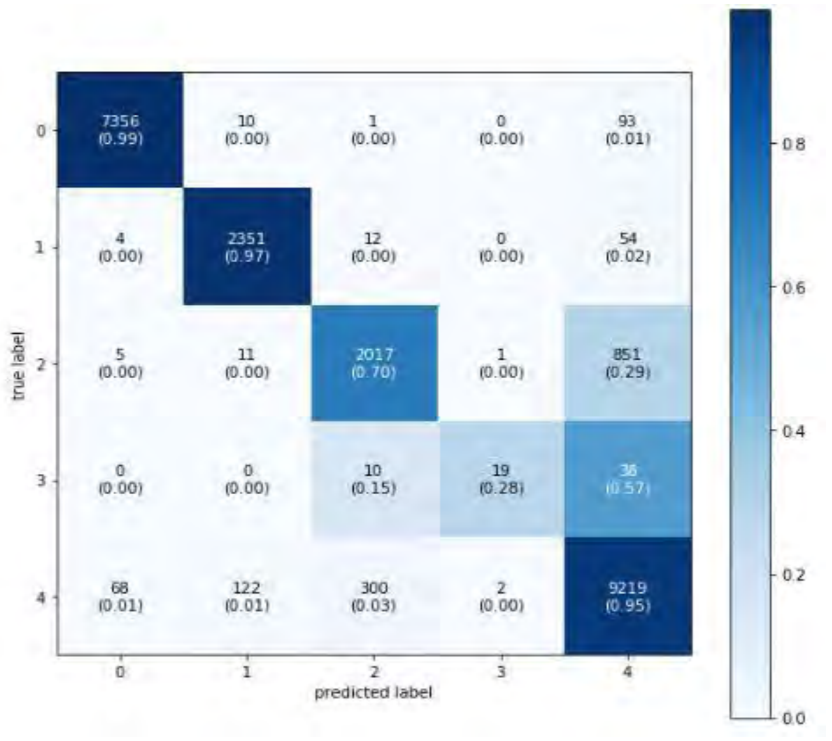


Figure 5.27: Confusion matrix of Stacking-2 (using 36 features)

Confusion Matrix of Stacking-3:

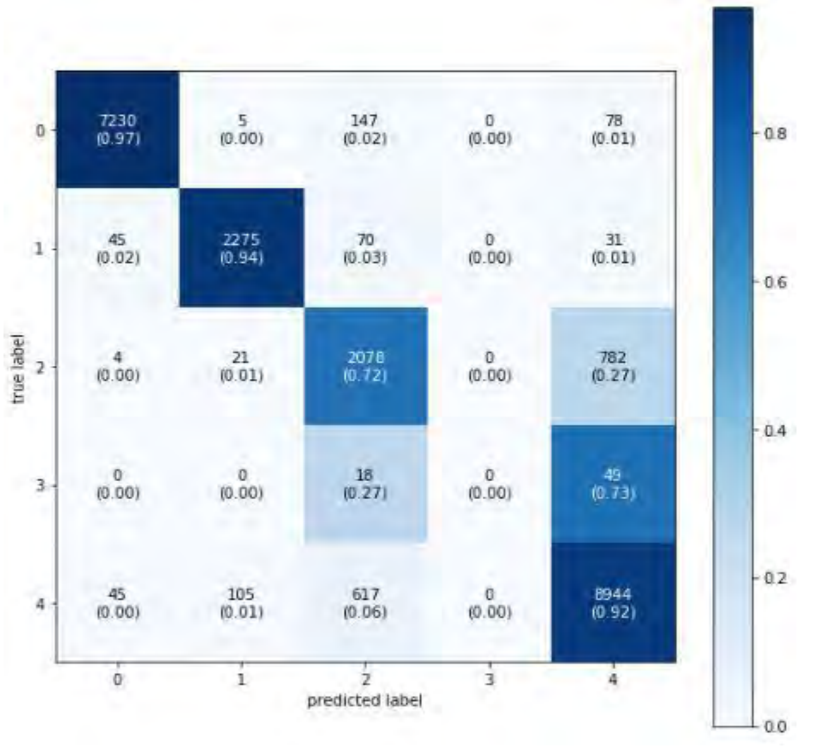


Figure 5.28: Confusion matrix of Stacking-3 (using 36 features)

Confusion Matrix of Stacking-4

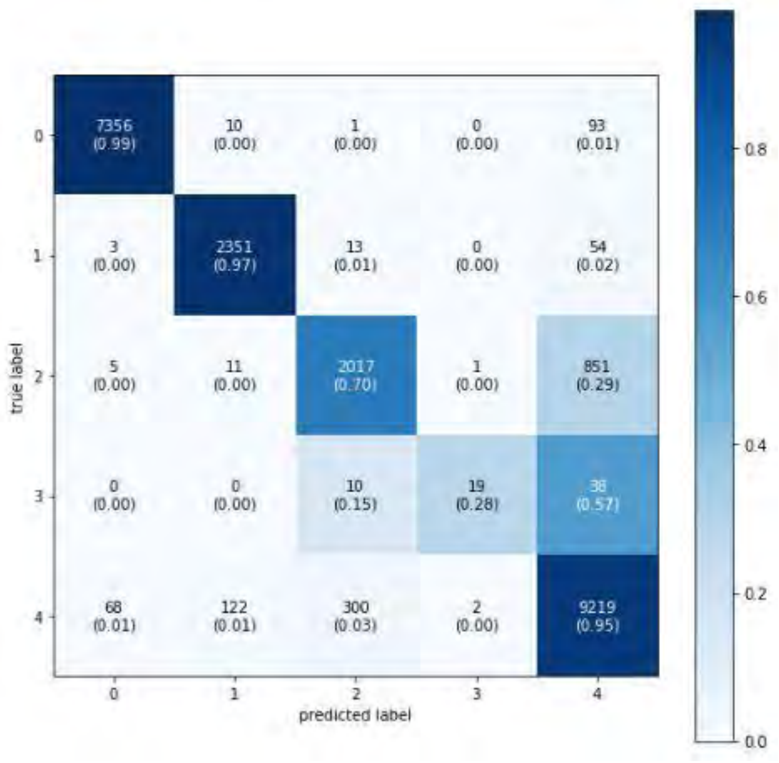


Figure 5.29: Confusion matrix of Stacking-4 (using 36 features)

5.2.2.2.1 Detection Rate of Stacking Methods

The performance in terms of detection rate of normal traffic and four types of attacks using four stacking classifiers has shown in Table 5.8. To clearly visualize, bar chart has depicted in Figure 5.30.

Table 5.8: Detection rate for Stacking with 36 features

Classifier	**Normal(4)**	**DoS(0)**	**Probe(1)**	**R2L(2)**	**U2R(3)**
Stacking -1	0.93	0.97	0.98	0.70	0.00
Stacking -2	0.95	0.99	0.97	0.70	0.28
Stacking -3	0.92	0.97	0.94	0.72	0.00
Stacking -4	0.95	0.99	0.97	0.70	0.28

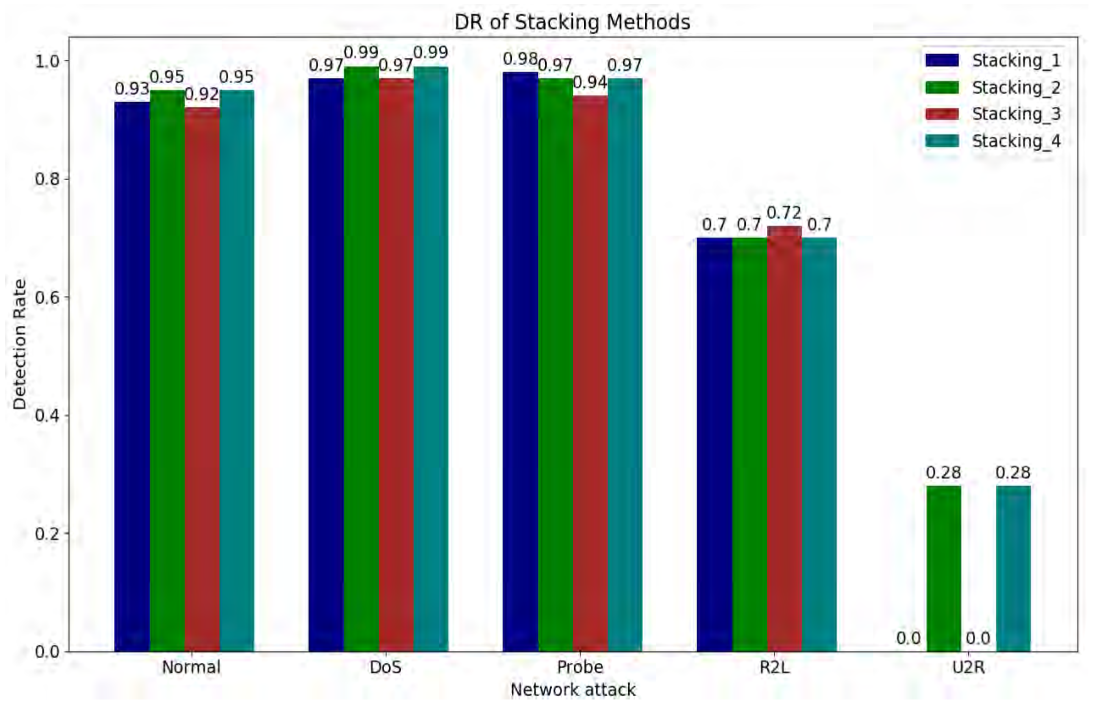


Figure 5.30: Detection rate for Stacking classifiers with 36 features

From Figure 5.30, it can be clearly seen that Stacking-2 and Stacking-4 perform same in every case and their detection rate is in satisfactory level. When use stacking-1, the DR of probe attack is the highest, in contrast when using stacking-3, the detection rate of R2l is the highest. However, Stacking-1 and Stacking-3 perform very poor in case of U2R attack.

5.2.2.3 Boosting

Boosting method discussed in section 3.6.2 and our proposed classifiers is shown in Table 3.1. In our thesis work, three type of Boosting methods are used, and their confusion matrices is depicted in Figure 5.31, Figure 5.32, Figure 5.34.

Confusion Matrix of Boosting-1

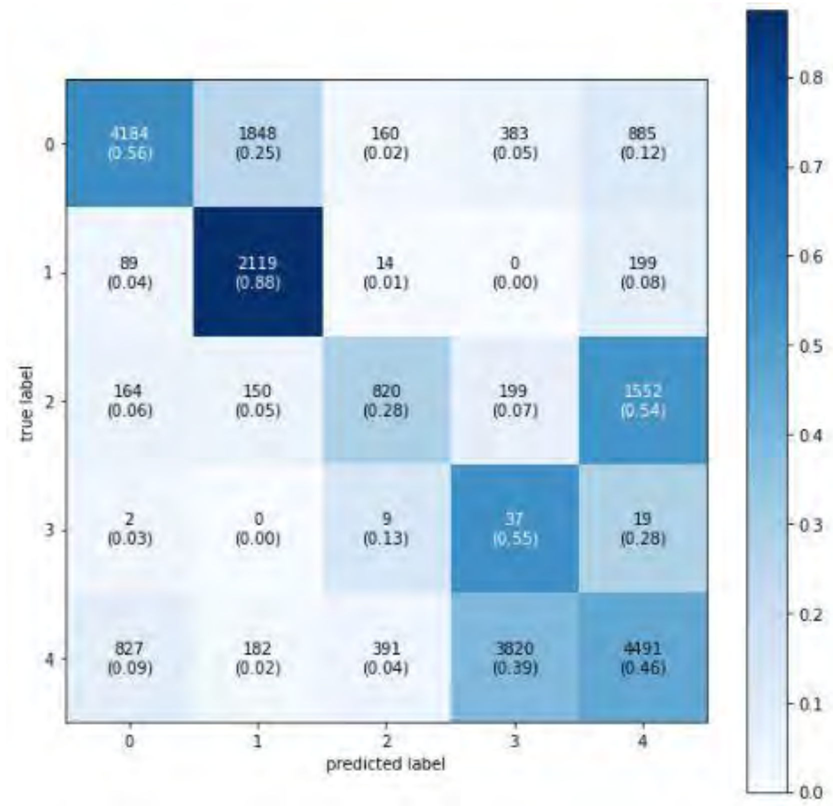


Figure 5.31: Confusion matrix of Boosting-1 (using 36 features)

Confusion Matrix of Boosting-2

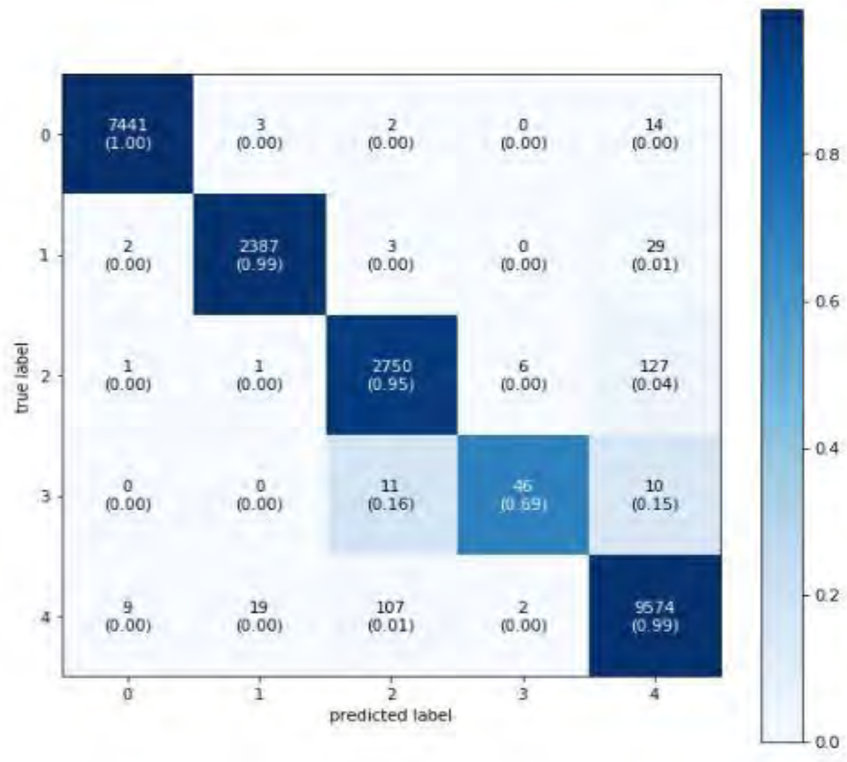


Figure 5.32: Confusion matrix of Boosting-2 (using 36 features)

ROC_AUC curve of Boosting-2

Since multiclass does not allow to plot ROC_AUC curve. Therefore, we sum up the DoS, Probe, R2L and U2R attacks and the combination named as Malware. After that, the ROC_AUC curve of Boosting-2 classifiers using normal traffic and malware traffic data is plotted in Figure 5.33.

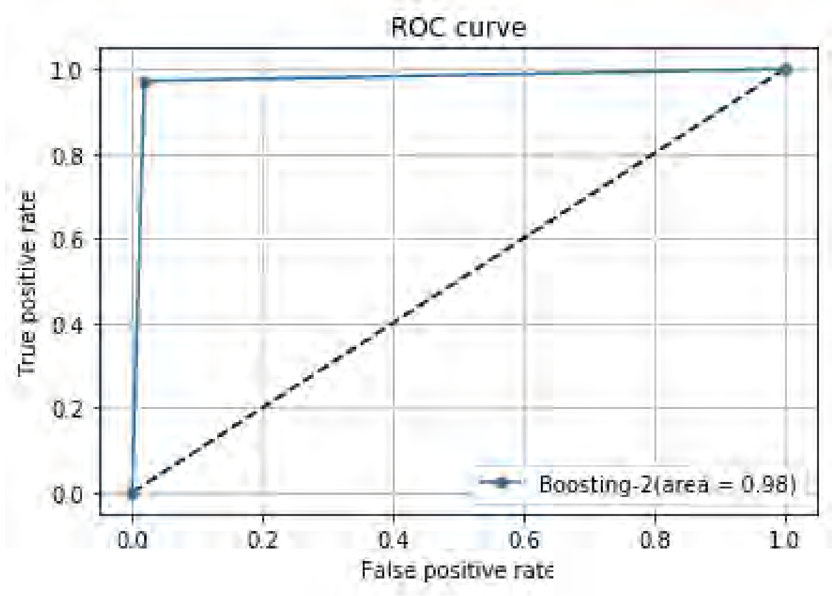


Figure 5.33: ROC_AUC of Boosting-2 classifier

Confusion Matrix of Boosting-3

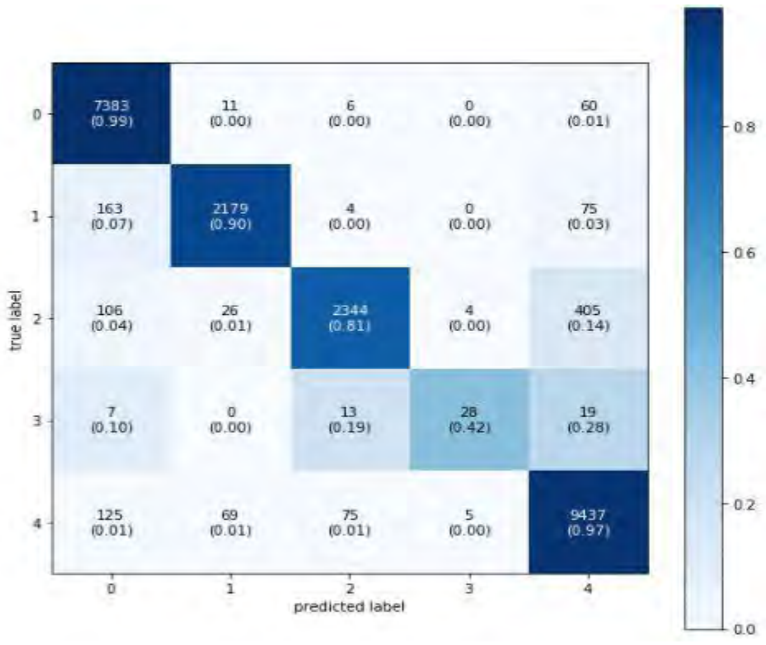


Figure 5.34: Confusion matrix of Boosting-3 (using 36 features)

5.2.2.3.1 Detection Rate of Boosting Methods

Detection rate of Boosting classifiers have shown in Table 5.9

Table 5.9: Detection rate for Boosting with 36 features

Classifier	Normal(4)	DoS(0)	Probe(1)	R2L(2)	U2R(3)
Boosting -1	0.46	0.56	0.88	0.28	0.55
Boosting -2	0.99	1.00	0.99	0.95	0.69
Boosting -3	0.97	0.99	0.90	0.81	0.42

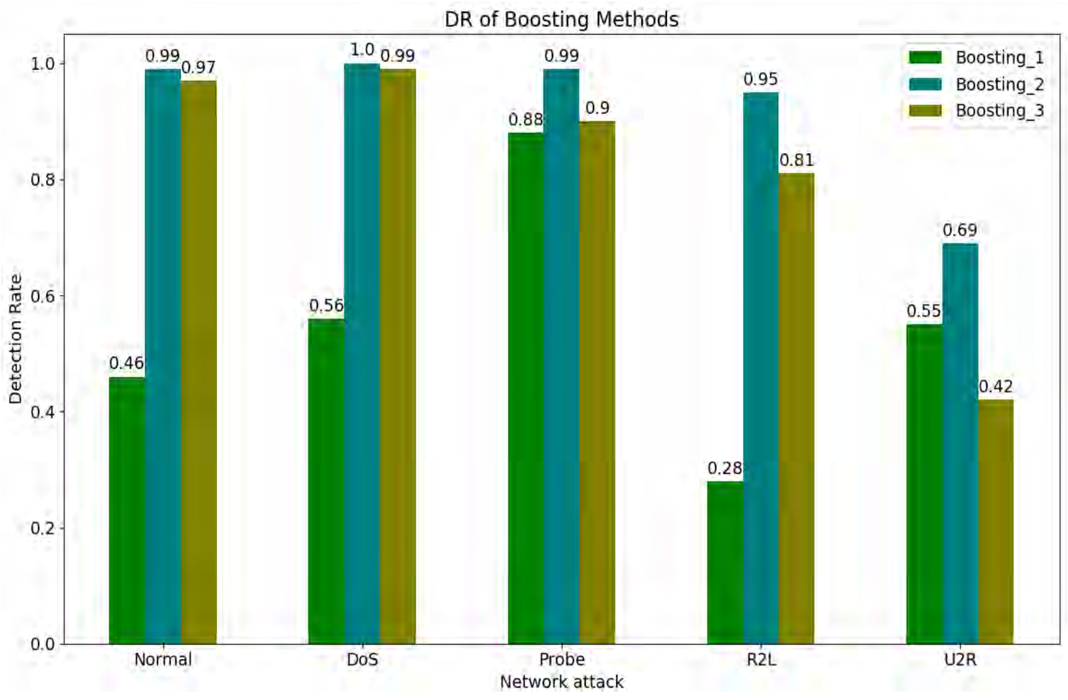


Figure 5.35: Detection rate for Boosting classifiers with 36 features

From Figure 5.35, it is clearly seen that the Boosting-2 classifier perform better in every case among other types of Boosting classifiers

5.2.2.4 Bagging

Bagging method is discussed in section 3.6.1. Three types of Bagging classifiers arrangement are shown in Table 5.10.

Table 5.10: three Bagging arrangements

Classifier	**Base Classifier**
Bagging-1	LR
Bagging-2	RandomForest
Bagging-3	Naïve Bayes

Confusion matrix of three types of bagging classifiers are shown in Figure 5.36, Figure 5.37 and in Figure 5.38

Confusion Matrix of Bagging-1

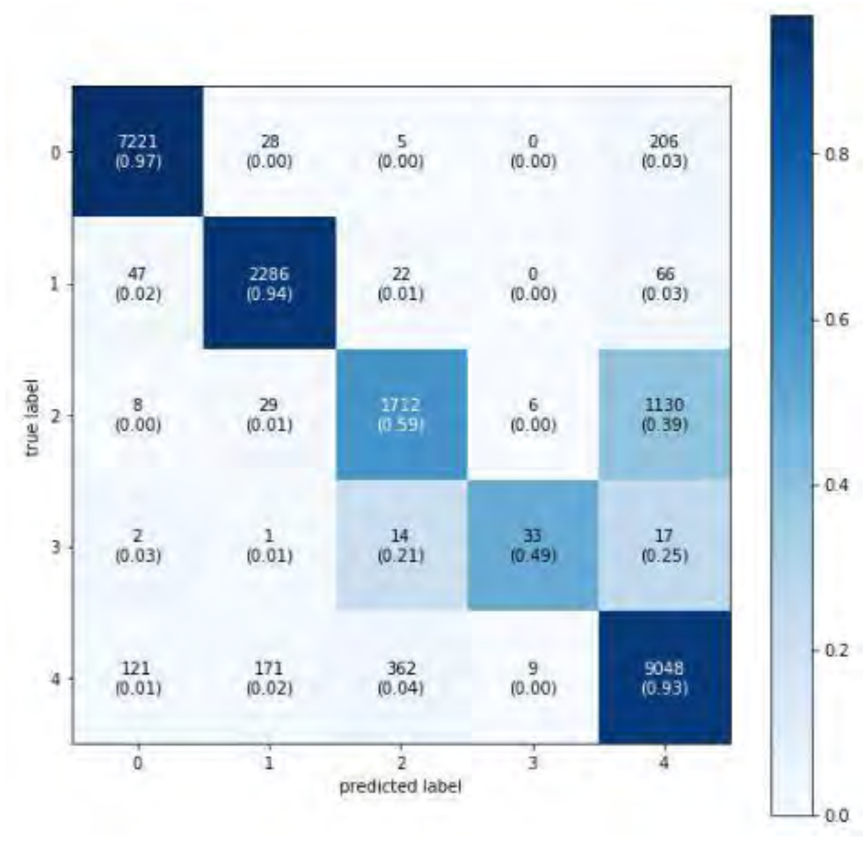


Figure 5.36: Confusion matrix of Bagging-1 (using 36 features)

Confusion Matrix of Bagging-2

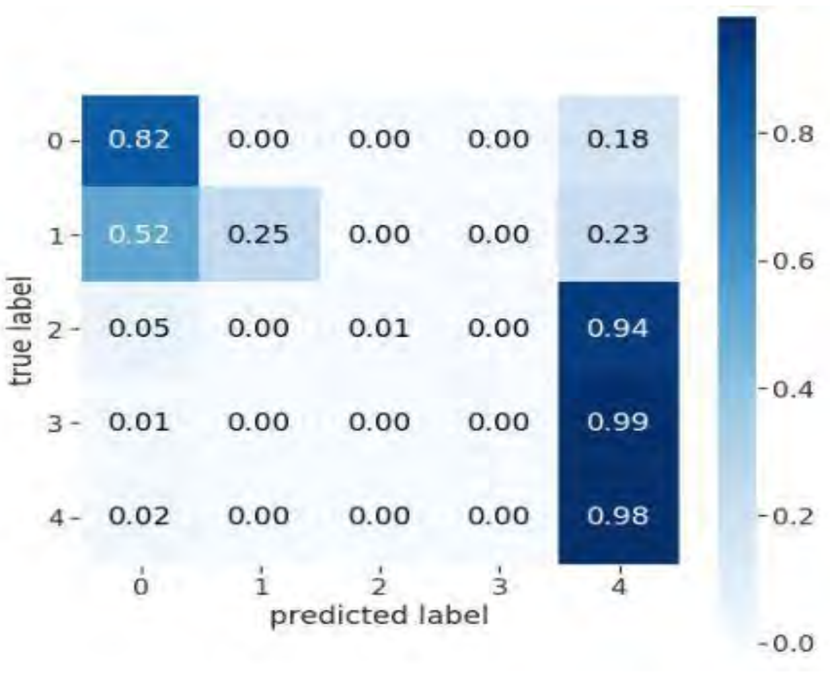


Figure 5.37: Confusion matrix of Bagging-2 (using 36 features)

Confusion Matrix of Bagging-3

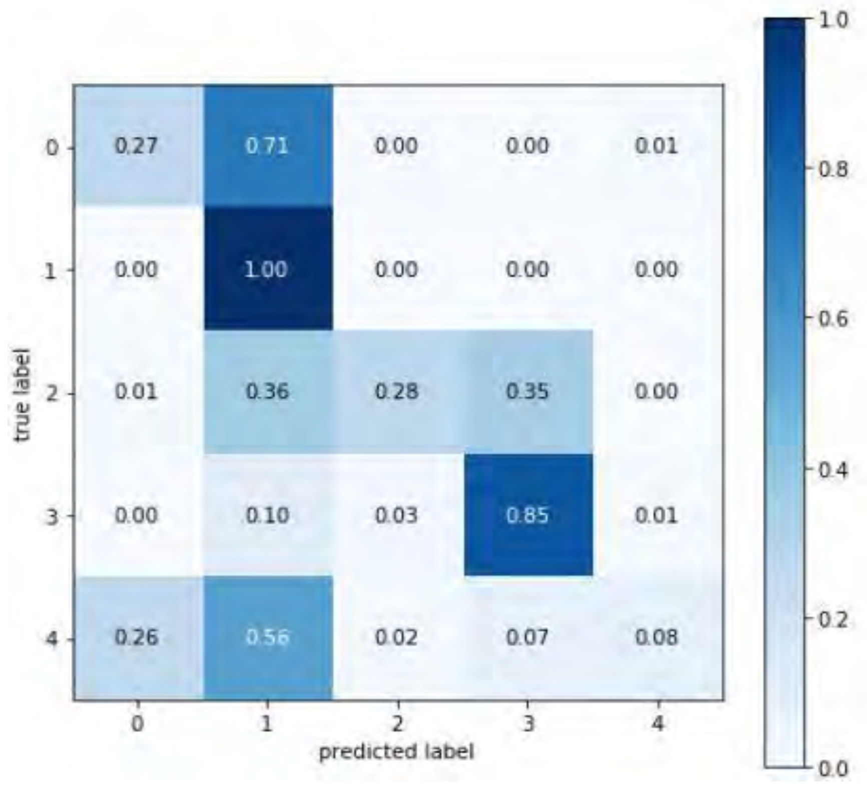


Figure 5.38: Confusion matrix of Bagging-3 (using 36 features)

5.2.2.4.1 Detection Rate of Bagging Methods

Detection rate using three types of Bagging classifiers is shown in Table 5.11

Table 5.11: Detection rate for Bagging with 36 features

Classifier	Normal(4)	DoS(0)	Probe(1)	R2L(2)	U2R(3)
Bagging -1	0.93	0.97	0.94	0.59	0.49
Bagging -2	0.98	0.82	0.25	0.01	0.00
Bagging -3	0.08	0.27	1.00	0.28	0.85

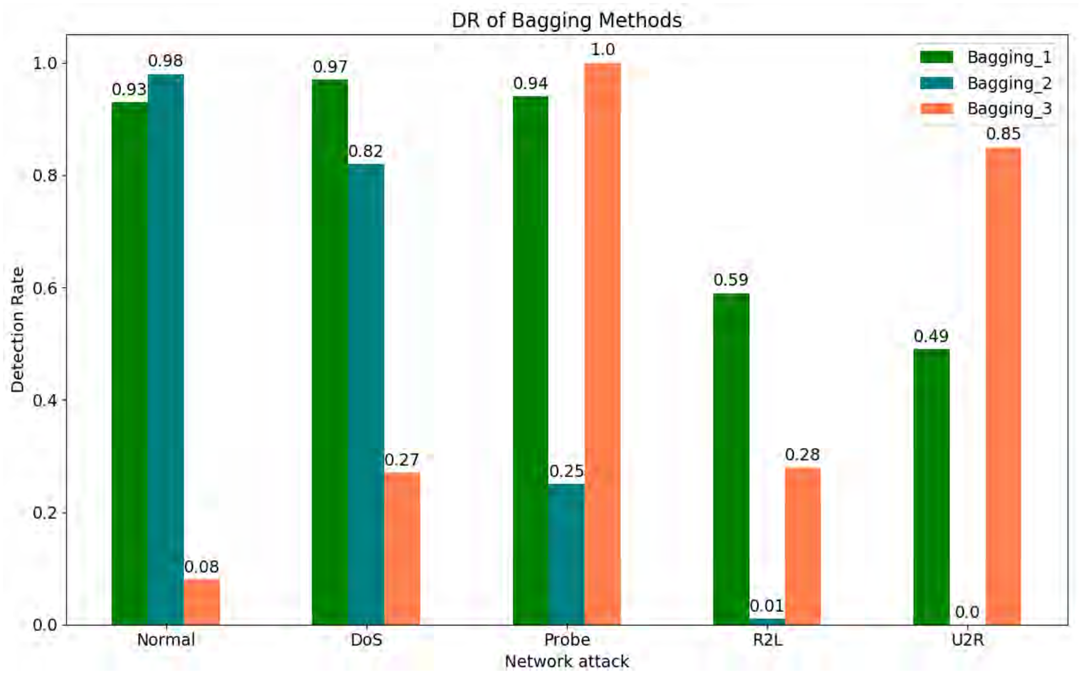


Figure 5.39: Detection rate of Bagging with 36 features

From Figure 5.39, Bagging-1 perform better in case of Normal traffic, Bagging-2 perform well in case of DoS and R2L attack detection and bagging-3 performs better in case of Probe and U2R attacks

5.2.2.5 Hybrid Model

Two hybrid models are shown in Table 5.12. For model-1 finally predict the result by voting method; however, as a base classifier it takes two ensemble methods (Adaboost and GradietBoosting) and one base level classifier (SVM). It is mainly a combination of Boosting and voting methods. Hybrid model-2 finally predicts by bagging algorithms, but it takes voting-1 as its base classifier which itself an ensemble classifier. It is mainly a combination of Voting and Bagging classifiers.

Table 5.12: hybrid models

Classifier	**description**
Hybrid-1 (Model-1)	- Base classifiers: Adaboost+ svc (kernel =RBF)+ GradietBoosting - Finally predicted by voting
Hybrid-2 (Model-2)	- Base Classifier: Voting-1 - Finally predicted by Bagging

Confusion matrix of model-1 and model-2 depicted in Figure 5.40 and in Figure 5.41 respectively

Confusion Matrix for Hybrid Model-1

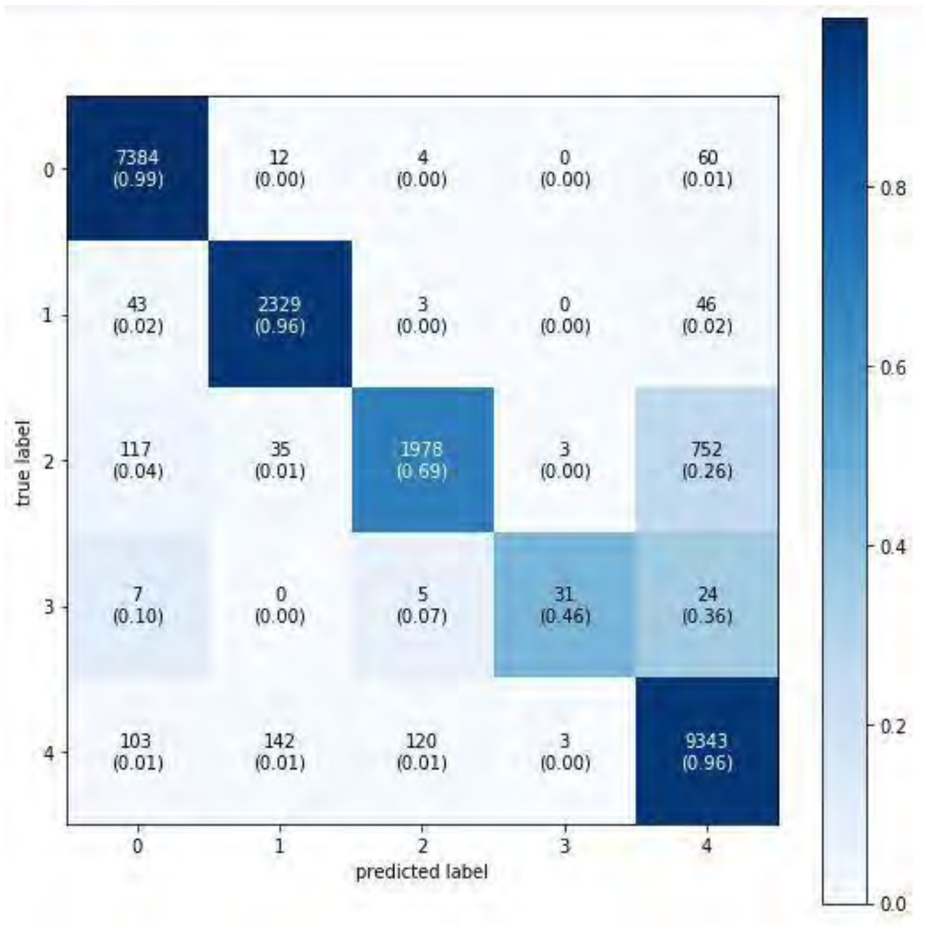


Figure 5.40: Confusion matrix of Model-1

Confusion Matrix for Hybrid Model-2

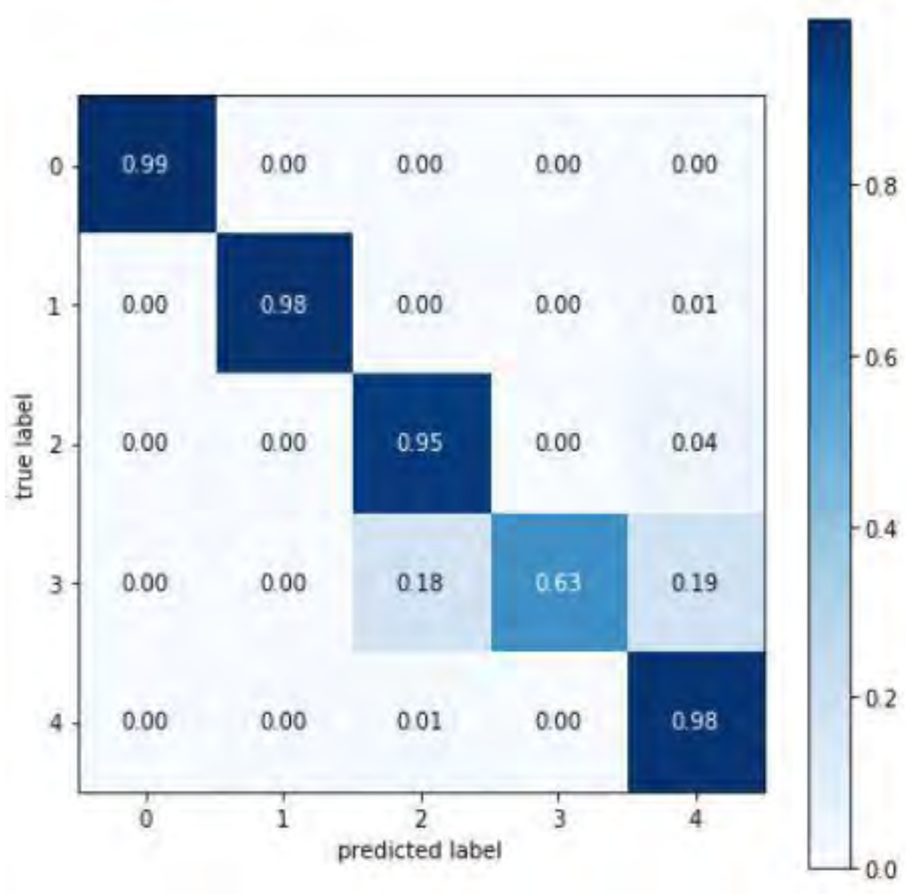


Figure 5.41: Confusion matrix of Model-2

5.2.2.5.1 Detection Rate of Hybrid Models

DR of every types of traffic shown in Table 5.13

Table 5.13: Detection rate for hybrid model with 36 features

Classifier	Normal(4)	DoS(0)	Probe(1)	R2L(2)	U2R(3)
Model -1 (Boosting + Voting)	0.96	0.99	0.96	0.69	0.46
Model -2 (Bagging + voting)	0.98	0.99	0.98	0.95	0.63

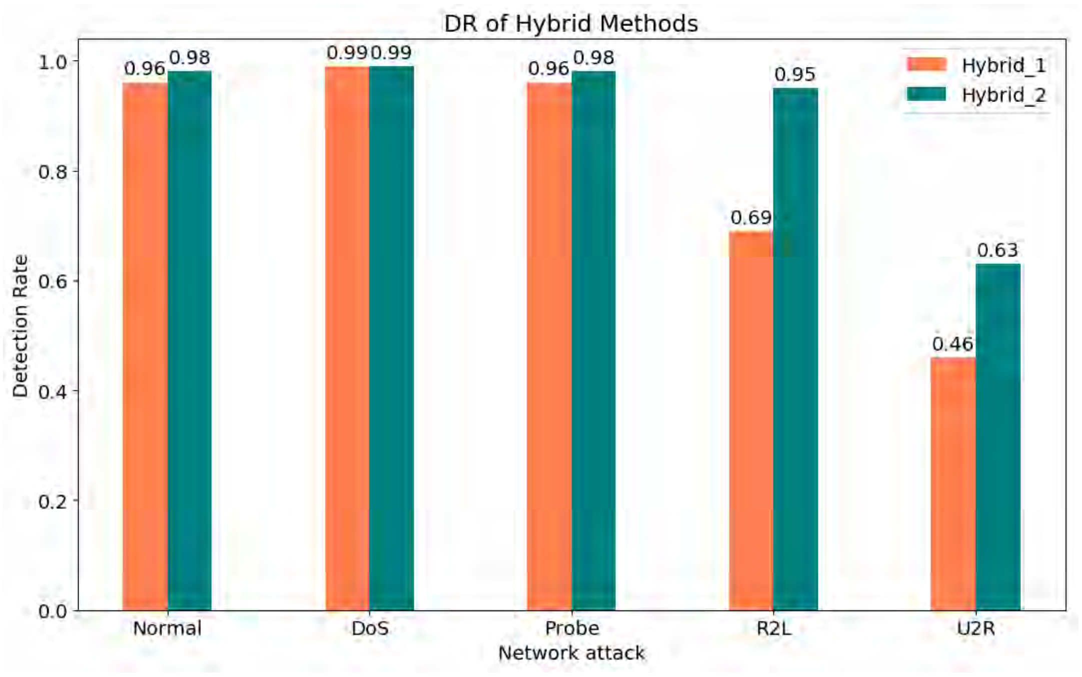


Figure 5.42: Detection rate for hybrid model with 36 features

From Figure 5.42, we can see that in every case, the Hybrid-2 performs better. Hybrid-2 mainly a Bagging of Voting-1. Now comparison of DR between Voing-1 and Hybrid-2 is shown in Table 5.14.

Table 5.14: Voting vs Bagging_voting

Classifier	**Normal(4)**	**DoS(0)**	**Probe(1)**	**R2L(2)**	**U2R(3)**
Voting-1	0.97	1.00	0.97	0.94	0.57
Hybrid-2 (Bagging_voting)	0.98	0.99	0.98	0.95	0.63

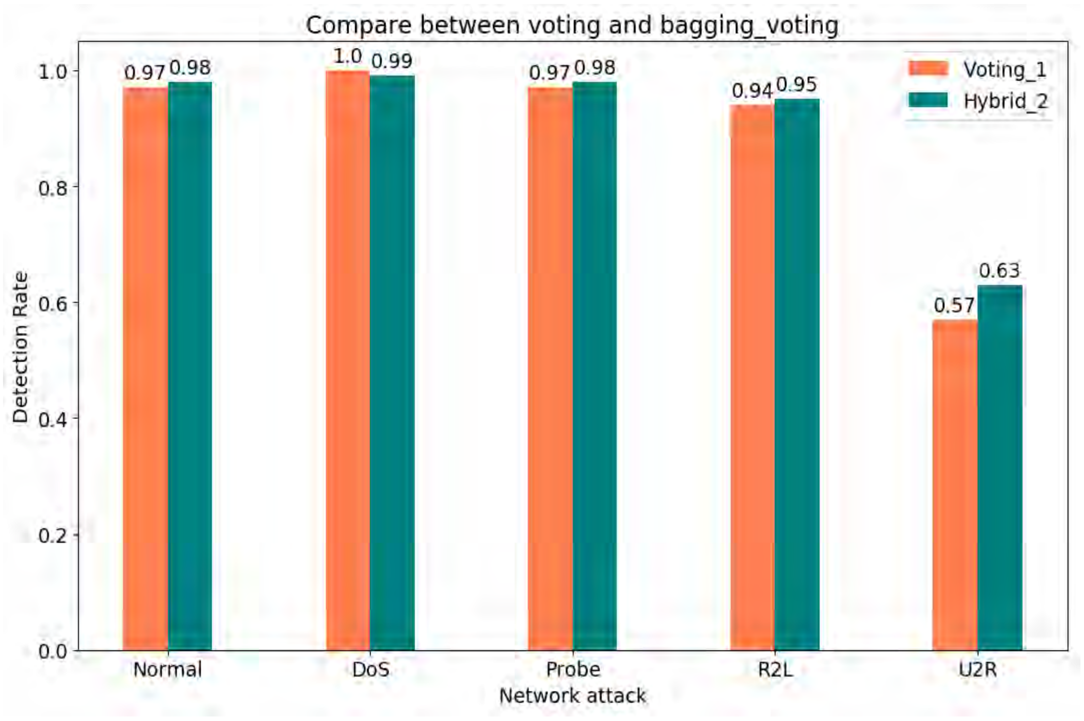


Figure 5.43: Voting vs Bagging_voting

It is clear from Figure 5.43 that bagging of voting-1 (Hybrid-2, Bagging+Voting) performs better than voting-1 except for DoS attack.

From Figure 5.25, we can see that voting-1 perform the best among four voting classifiers. Stacking-4 performs the best among four stacking classifiers (Figure 5.30). From figure 5.35, we can see that Boosting-2 performs the best to detect the network attacks. From figure 5.39, we can see that Bagging-1 performs the best to detect DoS and R2l attack; in contrast Bagging-3 performs the best to detect Probe and U2R attack. From Figure 5.42, we can see that the hybrid model-2 performs better than hybrid model-2 for detecting network attacks.

5.3 Performance Comparison of Classifiers

Comparison among 16 new designs of ensemble classifiers is shown in Figure 5.44, Figure 5.45, Figure 5.46, Figure 5.47 and in Figure 5.48.

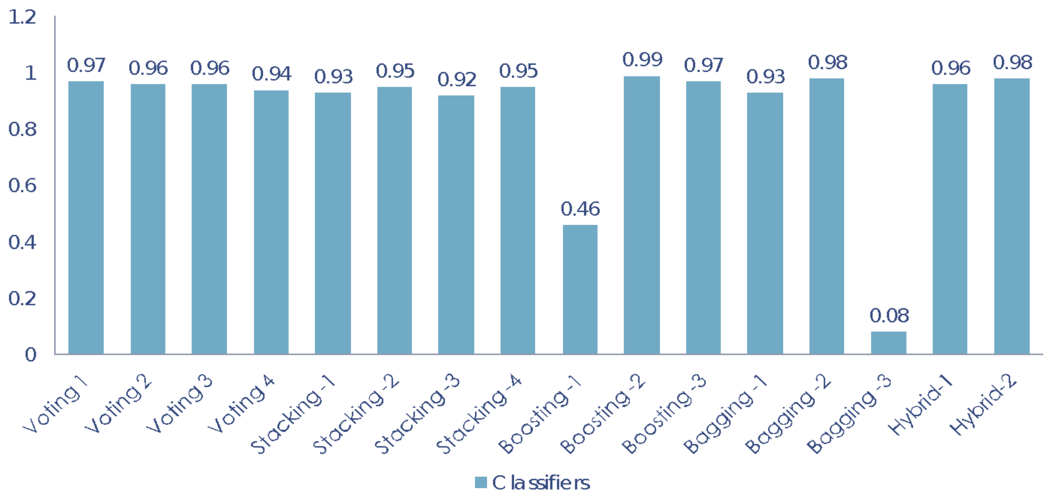


Figure 5.44: Comparison of normal traffic detection among 16 new arrangement of ensemble machine learning

From Figure 5.4, we can see that among 16 new design ensemble classifier Boosting-2 performs the best to detect normal traffic. Besides, Bagging-2 and Hybrid-2 classifiers perform better. Among classifiers, Bagging-3 performs the worst.

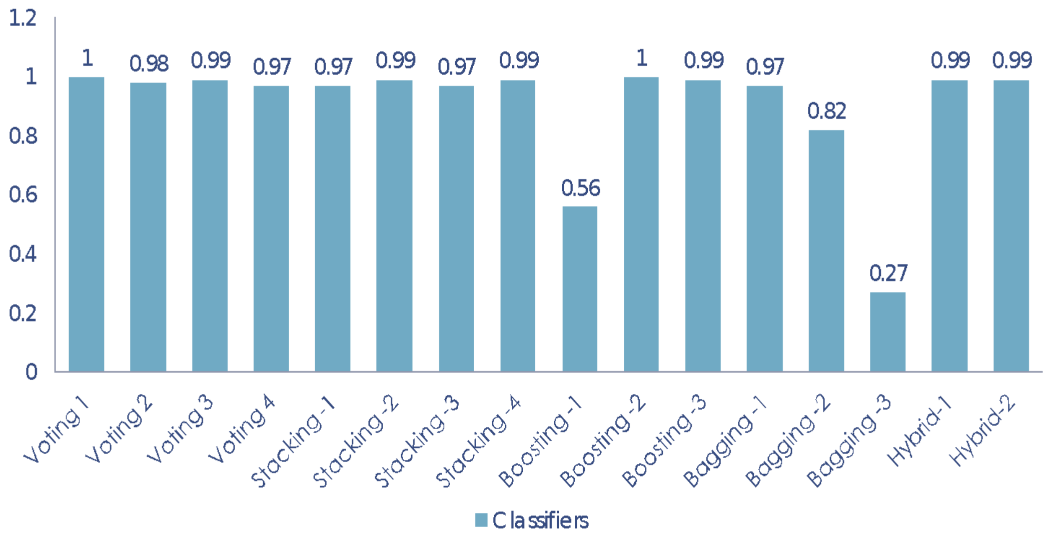


Figure 5.45: Comparison of DoS attack detection 16 new arrangement of ensemble machine learning

From Figure 5.45, we can see that to detect the DoS attack, voting-1 and Boosting-2 perform the best. Besides, Stacking-2, Stacking-4, Hybrid-1 and Hybrid-2 perform better. Among classifiers, Bagging-3 performs the worst.

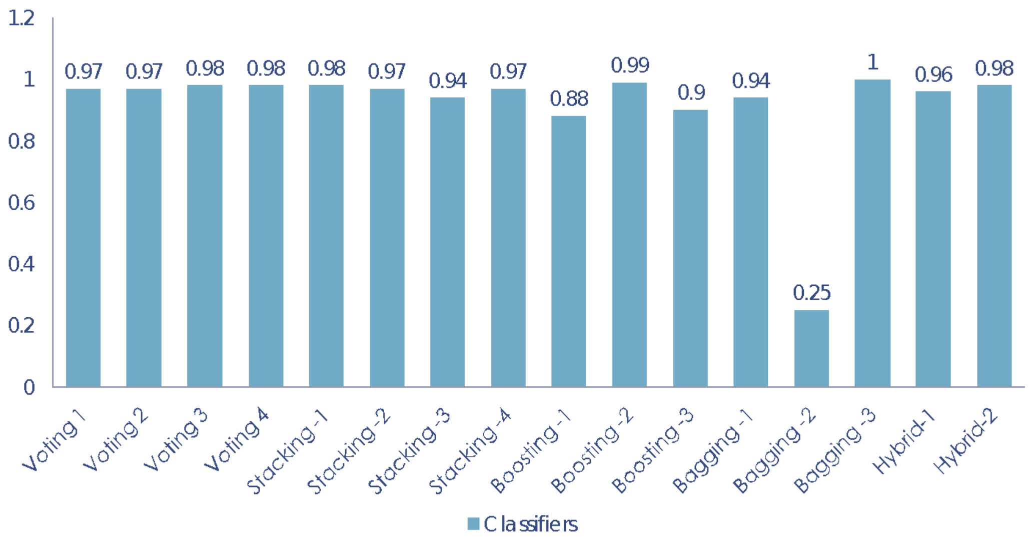


Figure 5.46: Comparison of Probe attack detection among 16 new arrangement of ensemble machine learning

To detect the probe attack, Bagging-3 performs the best. Besides, Boosting-2 performs better depicted in Figure 5.46. Among classifiers, Bagging-2 performs the worst.

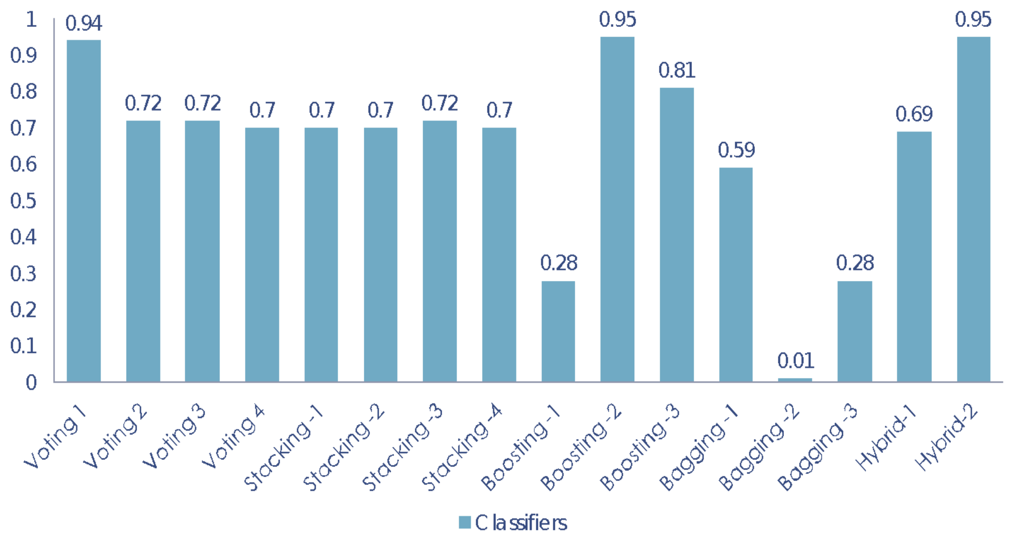


Figure 5.47: Comparison of R2L attack detection among16 new arrangement of ensemble machine learning

We can see from Figure 5.47 that Boosting-2 and Hybrid-2 performs the best to detect R2L attack whereas, the performance of Bagging-2 is the lowest.

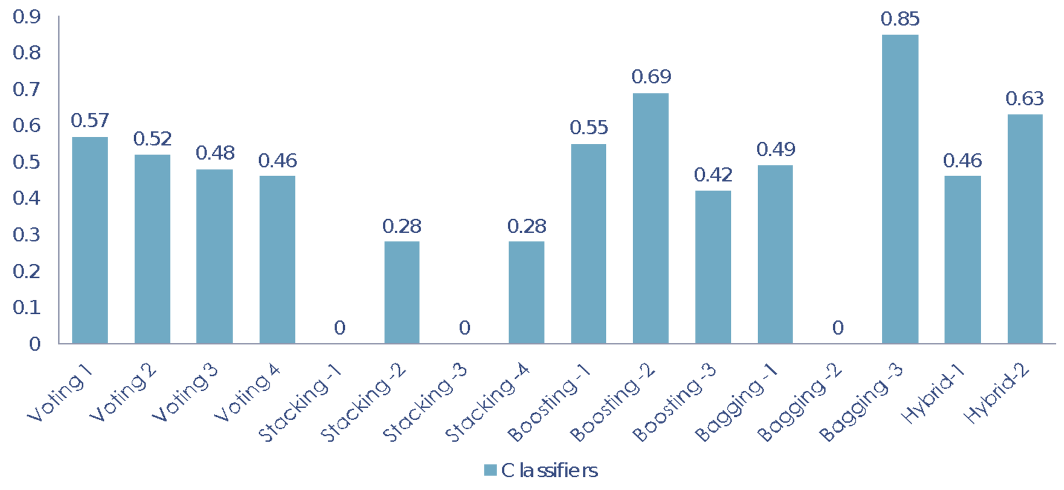


Figure 5.48: Comparison of U2R attack detection among 16 new arrangement of ensemble machine learning

Among 16 new design ensemble classifiers from Figure 5.48, Bagging-3 performs the best to detect U2R attack.

From Figure 5.44 to 5.48, we can see that for detecting the normal and DoS attack Bagging-3 perform the lowest. And Bagging-2 performs the lowest to detect the Probe, R2L and U2R attacks. The base class of Bagging-3 is Naïve Bayes which perform very worse when the data overlap each other, in contrast, the base class of Bagging-2 is a Tree-based classifier which does not bother when data is overlapping. For the first two cases, detecting normal and DoS attach Bagging-3 perform the worst due to the overlapping of normal and DoS attack data. On the other hand, for Probe, R2L and U2R the data does not overlap each other and Bagging-3 perform better. Due to not bother with data overlapping, Tree-base Bagging-2 performs opposite.

The all findings regarding detection rate of network attacks shown together in Table 5.15.

Table 5.15: Comparison of DR of normal traffic and four types of attacks among 16 new arrangement of ensemble machine learning

Classifiers	Normal	DoS	Probe	R2L	U2R
Voting 1	0.97	1.00	0.97	0.94	0.57
Voting 2	0.96	0.98	0.97	0.72	0.52
Voting 3	0.96	0.99	0.98	0.72	0.48
Voting 4	0.94	0.97	0.98	0.70	0.46
Stacking -1	0.93	0.97	0.98	0.70	0.00
Stacking -2	0.95	0.99	0.97	0.70	0.28
Stacking -3	0.92	0.97	0.94	0.72	0.00
Stacking -4	0.95	0.99	0.97	0.70	0.28
Boosting -1	0.46	0.56	0.88	0.28	0.55
Boosting -2	0.99	1.00	0.99	0.95	0.69
Boosting -3	0.97	0.99	0.90	0.81	0.42
Bagging -1	0.93	0.97	0.94	0.59	0.49
Bagging -2	0.98	0.82	0.25	0.01	0.00
Bagging -3	0.08	0.27	1.00	0.28	0.85
Hybrid-1 (Boosting of Voting-1)	0.96	0.99	0.96	0.69	0.46
Hybrid-2 (Bagging of voting-1)	0.98	0.99	0.98	0.95	0.63

From these research findings, it is clear that all of the features is not important for every types of network attacks. The detection rate can vary according to the number of features. Besides, every classifier does not perform well for every type of attack. The detection rate also can vary according to the choosing of classifier in every case.

Table 5.16 shows four types of attacks and proposed classifiers to detect those attacks separately and Table 5.17 shows the detection rate of one classifier which perform moderately to detect all types of attacks.

Table 5.16: Proposed classifiers for different types of network attacks

Network Attacks	**Proposed classifiers and DR**
DoS	Voting-1 (100%), Boosting-2(100%), Hybrid-1 (99%), Hybrid-2 (99%), Stacking-2 (99%), Stacking-4 (99%)
Probe	Bagging-3 (100%), Boosting-2 (99%)
R2L	Boosting-2 (95%), Hybrid-2(95%)
U2R	Bagging-3 (85%), Boosting-2 (69%)

Table 5.17 shows the detection rate of every types of attacks using Boosting-2 classifiers and we have used the first 36 important features to evaluate this model.

Table 5.17: Proposed classifiers for all types of attacks

Network Attacks	**DR (Boosting -2)**
DoS	100 %
Probe	99 %
R2L	95 %
U2R	69 %

5.4 Summary

According to the above discussion, we can see that different classifiers perform better for different types of attacks; however, we would like to propose a classifier which will perform moderately better for every type of attacks. We can see that the Boosting-2 classifier do this work better. Performance of this classifier is shown in Table 5.17.

Chapter 6
Conclusion and Future Works

6.1 Conclusion

In this thesis work, a novel approach has been proposed to predict intrusion in the network traffic based on packet header information that finally helps to develop effective classifiers. We used both without and with cross-validation to implement the ensemble classifiers. When we worked without cross validation, we used the full KSL-KDD training and testing dataset, and when we worked with-cross validation, we used the full KSL-KDD testing dataset. The proposed method without cross-validation achieves the improved detection rate in the case of R2L and U2R attacks compared to recent literature. In this paper, our main focus is on detection rate; therefore we compare our findings with recent literature that works on the detection rate using ensemble machine learning classifiers. For detecting normal and DoS attacks the detection rate of our proposed classifier is almost the same except Probe attack.

The proposed ensemble classifiers with cross validation have two parts. In the first part, after preprocessing, we do some feature engineering to get the best detection rate with reduces number of features. In the second part, we compare our 16 types of new design ensemble classifiers with each other. This work achieves detection rate of 100% for DoS attack, 100% for Probe attack, 95% for R2L attack and 85% for U2R attack. The proposed method starts working with a new set of features, which can be easily extracted from the network traffic. It introduces 36 features as a set to be used for the first time to detect the network attacks; those show impressive results and improve the performance of the proposed algorithm. Thus, the proposed algorithm proves to be better in terms of detection rate compared to other classifiers detecting network attacks.

6.2 Future Works

Despite its better performance than other similar algorithms, few limitations of the proposed algorithm have been found out during the thesis work. They will help to find out the scopes for further development of the algorithm. A list of these limitations is given below:

a) Not trying out platforms other than R, which could have led towards new discoveries by providing more control over the classifier parameters
b) Some other popular classifiers, such as deep learning with artificial neural networks and base level algorithms, could have been tested with
c) Architecture and data collection of intrusion detection that are not considered here, since the NSL-KDD data set is adopted in this work.
d) Using the detection rate of other algorithms reported in the literature, while comparing them with the newly proposed algorithm in this thesis work; as those algorithms could not be implemented due to lack of related information in the literature

Using the method detection rate of other algorithms reported by their authors in the literature, while comparing them with the newly proposed algorithm in this thesis work; as those algorithms could not be implemented due to lack of related information in the literature

To improve the algorithm functionality and provide additional evaluation of its performance, there are several areas of future work that can be served. They are:

a) Further improvements can be obtained by preparing data sets of higher quality. It should be possible to increase the number of data entries from updated databases.
b) Adding new network attacks or defining finer classifications will also be important in practical applications of predicting intrusion.
c) Other features of network traffic can be introduced and tested with

References

[1] Evans, D. (2011). The internet of things. How the Next Evolution of the Internet is Changing Everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG).

[2] A. Marx. 2008. Malware vs. Anti-Malware: (How) Can We Still Survive? Virus Bulletin, 2.2

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

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

[5] CNet news. Georgia accuses Russia of coordinated cyberattack, 2008. Available online: http://news. cnet.com/8301-1009_3-10014150-83.html.

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

[7] BBC. Cyber-security strategy launched, 2009a. Available online: http://news.bbc.co.uk/1/hi/uk_politics/8118348.stm.

[8] L. M. Lewis. A Case-Based Reasoning Approach to the Resolution of Faults in Communication Networks. In Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management with participation of the IEEE Communications Society CNOM and with support from the Institute for Educational Services, pages 671–682. North-Holland, 1993.

[9] 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.

[10] M. Sabhnani and G. Serpen. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. In *Proceedings of the International Conference on Machine Learning, Models, Technologies and Applications (MLMTA 2003)*, volume 1, pages 209–215, 2003.

[11] G. Cohen, M. Hilario, H. Sax, S. Hugonnet and A. Geissbuhler. 2006. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine*, 37, 7–18.

[12] M.A. Mazurowski, P.A. Habas, J.M. Zurada, J.Y. Lo, J.A. Baker and G.D. Tourassi. 2008. Training neural network classifiers for medical decision

making: The effects of imbalanced datasets on classification performance. *Neural Networks*, 21, 427–436.

[13] L. Mena and J.A. Gonzalez. Machine learning for imbalanced datasets: Application in medical diagnostic. In *Proceedings of the 19th International FLAIRS Conference*, 2006.

[14] Y-M. Huang, C-M. Hung and H.C. Jiau. 2006. Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. *Nonlinear Analysis: Real World Applications*, 7, 720–747.

[15] J. Burez and D. van den Poel. April 2009. Handling Class Imbalance in Customer Churn Prediction. *Expert Systems with Applications*, 36, 4626–4636.

[16] Y. Xie, X. Li, E.W.T. Ngai and W. Ying. 2009. Customer Churn Prediction using Improved Balanced Random Forests. *Expert Systems with Applications*, 36, 5445–5449.

[17] L. Kobyli´nski and A. Przepiórkowski. Definition Extraction with Balanced Random Forests. In *Advances in Natural Language Processing: Proceedings of the 6th International Conference on Natural Language Processing*, pages 237–247. Springer-Verlag, 2008.

[18] K. Kermanidis, M. Maragoudakis, N. Fakotakis and G. Kokkinakis. Learning Greek verb complements: addressing the class imbalance. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1065, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[19] E. Stamatatos. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing and Management*, 44, 790–799.

[20] N.V. Chawla. C4.5 and Imbalanced Data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *ICMLWorkshop on Learning from Imbalanced Datasets II*, 2003.

[21] T. Jo and N. Japkowicz. 2004. Class Imbalances versus Small Disjuncts. *SIGKDD Explorations Newsletter*, 6, 40–49.

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

[23] Y. Bouzida and F. Cuppens. Neural networks vs. decision trees for intrusion detection. In *IEEE / IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM)*, 2006b.

[24] A Javaid,Q Niyaz, Sun W, et al. "*A Deep Learning Approach for Network Intrusion Detection System*," Eai International Conference on Bio-Inspired Informa.

[25] F. Kuang,W. Xu, S. Zhang. *"A novel hybrid KPCA and SVM with GA model for intrusion detection,"* Applied Soft Computing Journal, 2014, 18(C):178-184.

[26] W. Li, P. Yi, Y. Wu et al. "*A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network"*. Journal of Electrical and Computer Engineering, 2014, 2014(5):1-8.

[27] B. Ingre, A. Yadav. *"Performance analysis of NSL-KDD dataset using ANN",* International Conference on Signal Processing and Communication Engineering Systems. IEEE, 2015:92-96.

[28] A. L. Buczak,E. Guven. "*A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection*," IEEE Communications Surveys and Tutorials, 2017, 18(2):1153-1176.

[29] N. Ben Amor, S. Benferhat and Z. Elouedi. Naive Bayes vs Decision Trees in Intrusion Detection Systems. In SAC '04: Proceedings of the 2004 ACM symposium on Applied computing, pages 420–424, New York, NY, USA, 2004. ACM. ISBN 1-58113-812-1.

[30] M. Panda and M.R. Patra. 2007. Network intrusion detection using naive bayes. IJCSNS International Journal of Computer Science and Network Security, 7, 258–263.

[31] F. Gharibian and A.A. Ghorbani. Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection. In CNSR '07: Proceedings of the Fifth Annual Conference on Communication Networks and Services Research, pages 350–358,Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2835-X.

[32] A. Bosin, N. Dessì and B. Pes. Intelligent Bayesian Classifiers in Network Intrusion Detection. In Proceedings of the 18th international conference on Innovations in Applied Artificial Intelligence, pages 445–447, London, UK, 2005. Springer-Verlag. ISBN 3-540-26551-1.

[33] D-P. Liu, M-W. Zhang and T. Li. Network Traffic Analysis Using Refined Bayesian Reasoning to Detect Flooding and Port Scan Attacks. In ICACTE

’08: Proceedings of the 2008 International Conference on Advanced Computer Theory and Engineering, pages 1000–1004, Washington, DC, USA, 2008a. IEEE Computer Society. ISBN 978-0-7695-3489-3.

[34] F. Qu,J. Zhang and Z. Shao. “An Intrusion Detection Model Based on Deep Belief Network,” Vi International Conference. 2017:97-101.

[35] J. Kim, J. Kim and H. T. L. Thu. *“Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection,”* International Conference on Platform Technology and Service. IEEE, 2016:1-5.

[36] B. Abolhasanzadeh. *“Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features,”* Information and Knowledge Technology. IEEE, 2015.

[37] U. Fiore, F. Palmieri and A. Castiglione. *“Network anomaly detection with the restricted Boltzmann machine,”* Neurocomputing, 2013, 122:13-23.

[38] Y. Ding, Y. Zhai, “*Intrusion Detection System for NSL-KDD Dataset Using Convolutional Neural Networks*,” The International Conference on Computer Science and Artificial Intelligence (CSAI), DOI: https://doi.org/10.1145/3297156.3297230, ISBN 978-1-4503-6606-9/18/12, ACM , 2018.

[39] Ali H. Mirza, *“Computer Network Intrusion Detection using various Classifiers and Ensemble Learning,”* 26th Signal Processing and Communications Applications Conference (SIU), IEEE, 2018

[40] Bo Hu, JinxiWang, Yifan Zhu and Tan Yang, *“Dynamic Deep Forest: An Ensemble Classification Method for Network Intrusion Detection,”* doi:10.3390/electronics8090968, Electronics 2019.

[41] Jivitesh Sharma1, Charul Giri1, Ole-Christoffer Granmo1 and Morten Goodwin, *“Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation,”* EURASIP Journal on Information Security, doi.org/10.1186/s13635-019-0098-y, 2019.

[42] P.G. Neumann and P.A.. Porras. Experience with EMERALD to Date. In *First USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 73–80, Santa Clara, California, apr 1999.

[43] 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.

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

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

[46] "ids 2017 | datasets | research | canadian institute for cybersecurity | unb." [online]. available: http://www.unb.ca/cic/datasets/ids-2017.html. [accessed: 12- oct-2019].

[47] M. Asaka, A. Taguchi and S. Goto. The Implementation of IDA: An Intrusion Detection Agent System. In Proceedings of the 11th Annual FIRST Conference on Computer Security Incident Handling and Response (FIRST'99), 1999.

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

[49] K. Kendall. A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Master's thesis, Massachusetts Institute of Technology, 1999.

[50] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham and M. Zissman. Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In Proceedings of the DARPA Information Survivability Conference and Exposition, volume 2, pages 12–26. IEEE Computer Society Press, 2000a.

[51] R. Verwoerd and R. Hunt. September 2002. Intrusion Detection Techniques and Approaches. Computer Communications, 25, 1356–1365.

[52] D. Gollmann. *Computer Security*. Wiley, 2nd edition, 2006.

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

[54] W. Lee and D. Xiang. Information Theoretic Measures for Anomaly Detection. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.

[55] H. Debar, M. Dacier and A. Wespi. 1999. Towards a Taxonomy of Intrusion Detection Systems. *Computer Networks*, 31, 805–822.

[56] H. Debar. An Introduction to Intrusion Detection Systems. In *Connect 2000*, 2000.

[57] D.E. Denning. 1987. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13, 222–232. ISSN 0098-5589.

[58] D. Endler. Intrusion Detection: Applying Machine Learning to Solaris Audit Data. In Proceedings of the Annual Computer Security Applications conference, pages 267–279, 1998.

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

[60] M. Panda and M.R. Patra. 2007. Network intrusion detection using naive bayes. *IJCSNS International Journal of Computer Science and Network Security*, 7, 258–263.

[61] M. Panda and M.R. Patra. Ensemble of classifiers for detecting network intrusion. In *ICAC3 '09: Proceedings of the International Conference on Advances in Computing, Communication and Control*, pages 510–515, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-351-8.

[62] C. Xiang, P.C. Yong and L.S. Meng. 2008. Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees. *Pattern Recogn. Lett.*, 29, 918–924. ISSN 0167-8655.

[63] J. Zhang and M. Zulkernine. A Hybrid Network Intrusion Detection Technique Using Random Forests. In *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*, pages 262–269, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2567-9.

[64] L. M. Lewis. A Case-Based Reasoning Approach to the Resolution of Faults in Communication Networks. In *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management with participation of the IEEE Communications Society CNOM and with support from the Institute for Educational Services*, pages 671–682. North-Holland, 1993.

[65] "network intrusion detection signatures, part one | symantec connect community." [online]. available: https://www.symantec.com/connect/articles/networkintrusion- detection signatures-part-one.

[66] "ids: Signature versus anomaly detection," searchsecurity. [online]. available: https://searchsecurity.techtarget.com/tip/ids-signature-versus-anomalydetection.

[67] "sans: Website security." [online]. available: https://www.sans.org/security/. [Accessed 21 December 2019]

[68] D. Zhang M. Zhang H. Li, Y. Wang and E. Y. Chang. "pfp: parallel fp-growth for query recommendation," in proceedings of the 2008 acm conference on recommender systems - recsys '08, lausanne, switzerland, 2008, p. 107.

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

[70] "promiscuous mode (linktionary term)." [online]. available: http://www.linktionary.com/p/promiscuous.html.

[71] Zhang Chao-Yang. Dos attack analysis and study of new measures to prevent. 2011 International Conference on Intelligence Science and Information Engineering, 2011.

[72] Jeremy Seth Davis. Sony psn downed; hacking group claims ddos attack | sc media, Jul 2018.

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

[74] "introduction — bro 2.5.5 documentation." [online]. available: https://www.bro.org/sphinx/intro/index.html.

[75] Jparaiso. "cisco - netranger intrusion detection system.", Dec 1998.

[76] S. Hua and Z. Sun, "Support vector machine approach for protein subcellular localization prediction," *Bioinformatics,* vol. 17, no. 1, pp. 721-728, 2001.

[77] T. H. Lin, R. F. Murphy and Z. Bar-Joseph , "Discriminative motif finding for predicting protein subcellular localization," *IEEE/ACM Transaction of Computational Biology and Bioinformatics,* vol. 8, no. 2, pp. 441-451, 2011.

[78] R. N. Kalate, S. S. Tambe and B. D. Kulkarni, "Artificial neural networks for prediction of mycobacterial promoter sequences," *Computational Biology and Chemistry,* vol. 27, no. 6, pp. 555-564, 2003.

[79] X. Xiao, Z.-C. Wu and K.-C. Chou, "A multi-label classifier for predicting the subcellular localization of gram-negative bacterial proteins with both single and multiple sites," *PLoS ONE,* vol. 6, 2011.

[80] P. Shah, "Insights into Machine Learning," 8 1 2018. [Online]. Available: https://opensourceforu.com/2018/01/insights-machine-learning/.

[81] P. Hassani, "An Insight into 26 Big Data Analytic Techniques: Part 2," 30 11 2016. [Online]. Available: https://blogs.systweak.com/2016/11/an-insight-into-26-big-data-analytic-techniques-part-2/. [Accessed 21 December 2019]

[82] Priyadharshini, "Machine Learning: What it is and Why it Matters," 18 3 2018. [Online]. Available: https://www.simplilearn.com/what-is-machine-learning-and-why-it-matters-article.

[83] R. van Loon, "Machine Learning Explained: Understanding Supervised, Unsupervised, and Reinforcement Learning," 6 1 2018. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/machine-learning-explained-understanding-supervised-unsupervised.

[84] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2), 121–167.

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

[86] T. Hill and P. Lewicki, "Support Vector Machines," in *Electronic Statistics Textbook*, StatSoft Inc. 1995.

[87] C. M. Bishop, Pattern Recognition and Machine Learning, Springer.

[88] B. Yekkehkhany, A. Safari, S. Homayouni and M. Hasanlou, "A Comparison Study of Different Kernel Functions for SVM-based Classification of," *The International Archives of the photogrammetry, Remote Sensing and Spatial Information Sciences,* 2014.

[89] Y. Liu and K. K. Parhi, "Computing RBF kernel for SVM classification using stochastic logic," *Proceedings - IEEE International Workshop on Signal Processing Systems, SiPS 2016,* pp. 327-332.

[90] M. Ring and B. M. Eskofier, "An approximation of the Gaussian RBF kernel for efficient classification with SVMs," *Pattern Recognition Letters,* vol. 84, no. C, pp. 107-113, 2016.

[91] C.-W. Hsu, C.-C. Chang, C.-J. Lin et al., "A practical guide to support vector classification," 2003.

[92] Keerthi, S. S., & Lin, C. J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation, 15(7), 1667–1689.

[93] Lin, H. T., and Lin, C. J. (2003). A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Taipei: Department of Computer Science and Information Engineering, National Taiwan University.

[94] R. Jin, F. Yan and J. Zhu, Application of Logistic Regression Model in an Epidemiological Study, *Science Journal of Applied Mathematics and Statistics*, **3** (2015), no. 5, 225-229. https://doi.org/10.11648/j.sjams.20150305.12

[95] V. Losing, B. Hammer and H. Wersing, "KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift," in *IEEE 16th International Conference on Data Mining (ICDM)*, 2016.

[96] A. Bronshtein, "A Quick Introduction to K-Nearest Neighbors Algorithm," 2017. [Online]. Available: https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7.

[97] https://www.datacamp.com/community/tutorials/ensemble-learning-python [Accessed 21 December 2019]

[98] https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/ [Accessed 21 December 2019]

[99] https://medium.com/@rrfd/boosting-bagging-and-stacking-ensemble-methods-with-sklearn-and-mlens-a455c0c982de [Accessed 21 December 2019]

[100] Sapna S. Kaushik, Dr. Prof.P.R.Deshmukh," Detection of Attacks in an Intrusion Detection System", International Journal of Computer Science and Information Technologies, Vol. 2 (3), 2011, 982-986

[101] http://nsl.cs.unb.ca/NSL-KDD/ [Accessed 21 December 2019]

[102] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani "A Detailed Analysis of the KDD CUP 99 Data Set", Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)

[103] S. Velliangiri and J. Premalatha. Intrusion detection of distributed denial of service attack in cloud. Cluster Computing, Apr 2017.

[104] Isabelle Guyon and Andr Elisseeff. 2003. An introduction to variable and feature selection. *Jouranl of Machine Learning Research* 3 (March 2003), 1157–1182.

[105] Randall Wald, Taghi M. Khoshgoftaar, and Amri Napolitano. 2013. Comparison of stability for different families of filter-based and wrapper-based feature selection. In *ICMLA*.

[106] Alain Rakotomamonjy. 2003. Variable selection using SVM-based criteria. *JMLR* 3 (March 2003), 1357–1370.

[107] Simon Perkins, Kevin Lacker, and James Theiler. 2003. Grafting: Fast incremental feature selection by gradient descent in function space. *JMLR* 3 (March 2003), 1333–1356.

[108] Yvan Saeys, Inaki Inza, and Pedro Larranaga. 2007. A review of feature selection techniques in bioinformatics. *Bioinformatics* 23, 19 (2007), 2507–2517.

[109] https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection [Accessed 21 December 2019]

[110] “scikit-learn: machine learning in python — scikit-learn 0.20.0 documentation.” [online]. available: http://scikit-learn.org/stable/. [Accessed 21 December 2019]

[111] Xianwei Gao , Chun Shan , Changzhen Hu, Zequn Niu , And Zhen Liu, *“An Adaptive Ensemble Machine Learning Model for Intrusion Detection,”* doi.10.1109/ACCESS.2019.2923640, IEEE Access, June 19, 2019.

Appendix A

A.1

The first few pages of accepted thesis proposal

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
OFFICE OF THE MEMBER SECRETARY OF THE COMMITTEE FOR ADVANCED STUDIES & RESEARCH, BUET, DHAKA.

Application form (Guide lines) for approval of M.Sc. in ICT. Thesis proposal by the CASR. All the items, which are applicable of the following list, must be mentioned and filled in properly. You can submit your soft copy to Email: casr@daers.buet.ac.bd

Date : 18/11/2019

1.	**Name of the student**: MD. RAIHAN-AL-MASUD	**Status:** Full-Time
	Roll No. 1015312017	**Session:** October, 2015
2.	**Present Address:** 372/6, South Kafrul, Dhaka Cantt	**Cell:** 01723336143
3.	**Name of the Dept/Inst.:** IICT	**Programme** : M.Sc. in ICT
4.	**Name of the Supervisor:** Dr. Hossen Asiful Mustafa	**Designation**: Assistant Professor
5.	**Name of the Co-Supervisor (if any):** N/A	**Designation** :N/A
6.	**Date of First Enrolment in the Program:** OCTOBER-2015	

7. **Tentative Title:** AN ENSEMBLE MACHINE LEARNING APPROACH FOR NETWORK INTRUSION DETECTION

8. **Background and present state of the problem:**

Network Intrusion Detection Systems (NIDSs), particularly, Anomaly Detection Systems (ADSs) [1], have become more significant in detecting novel attacks than Signature Detection Systems (SDSs) [2]. One of the key intrusion detection systems is to inspect the network traffic flow between hosts, and the network packets to discriminate between the observations: normal or abnormal. Currently, due to the massive growth in computer networks and applications, many challenges arise for network security research. Intrusions/attacks are able to compromise the principles of computer systems, e.g., availability, authority, confidentiality and integrity. Firewall systems cannot detect modern attack environments and are not able to analyze network packets in depth. A NIDS monitors network traffic flow to identify attacks. The signature-based NIDS matches the existing of known attacks to detect intrusions. However, in the anomaly-based NIDS, a normal profile is created from the normal behavior of the network and any deviation from this is considered as an attack. Further, the signature-based NIDSs cannot detect unknown attacks, and for these anomalies, anomaly-based NIDS are recommended in many studies [3] [4]. In this context, anomaly-based network intrusion detection techniques are a valuable technology to protect target systems and networks against malicious activities. However, despite the variety of such methods described in the literature [5-9] in recent years, security tools incorporating anomaly detection functionalities are just starting to appear, and attack detection accuracy still not good enough. Researchers have used different machine learning methodologies for NIDS but ensemble machine learning approach has not been applied effectively. Ensemble method combines several base models in order to produce one optimal predictive model. Additionally, considering the inclusion of large number of features and complexity of the algorithms with respect to the classification, further improvement is required in classifier design and feature engineering for better performance in malicious network traffic classification.

9. (a) Objectives with specific aims:

The objective of this thesis is to develop an algorithm using ensemble machine learning techniques for better detection of malware in the network traffic. To achieve this objective, the following specific aims are identified.

1. To identify an order of importance of features for effective detection of malware by packet header inspection found within network traffic.
2. To design an efficient classifier algorithm for identifying malware in target networks based on their features.
3. To implement the algorithm for simulation and compare the performance with existing works in the literature.

(b) Possible outcome:

Successful completion of this research will result in a machine learning algorithm which can detect malware in the network traffic data with high accuracy.

10. Outline of Methodology/ Experimental Design:

In this research, standard network packet header information available on the web for normal traffic will be considered and 4 different types of attacks such as Denial of Service (DoS), Probe, Remote to Local (U2R) and User to Root (R2L) in the network will be addressed. Below is a list of activities which will be followed throughout this research:

i. An order of importance of features will be derived from the packet header information for ease of the feature computation and simplicity of the method. Feature significance will be studied through hypothesis testing and computing parameters such as p-value.

ii. An Ensemble machine learning classification algorithm based on Voting, Stacking, Bagging and Boosting will be developed considering its wide range of applicability and superior performances for intrusion detection in network traffic compared to other algorithms.

iii. A Python based library Scikit Learn will be used for pre-processing, feature engineering, classification algorithm design, training and testing to develop the algorithms based on ensemble machine learning algorithms to improve the security in a network intrusion detection system.

iv. The performance of the algorithm (e,g., sensitivity or detection rate) will be evaluated on independent training and testing dataset and will be compared with existing other algorithms.

v. Then, a detailed analysis will be performed to find the suitability of the available machine learning algorithms for intrusion detection system. Best performing machine learning algorithms including base level[6] as well as deep learning[7-9] on literature will be explored for suitability (e. g., Support Vector machine (SVM) [6], Discriminative Restricted Boltzmann Machine (DBM) [7], long short-term memory (LSTM) [8], Convolutional neural network (CNN))[9].

vi. The Strength of the proposed system will be shown in terms of area under of receiver operating characteristics (ROC-AUC) curve, sensitivity, specificity as well as Precision-Recall curve (PR-Curve).

11. References:

[1] Bhuyan, H. M., Bhattacharyya, K. D., Kalita, K. J., "Network Anomaly Detection: Methods, Systems and Tools," *IEEE Communication Surveys and Tutorials*, 2013.

[2] Mujumdar, A., Masiwal, G., Meshram, B. B., "Analysis of Signature-Based and Behavior-Based Anti-Malware Approaches," *International Journal of Advanced Research in Computer Engineering and Technology*, 2013.

[3] Dartigue, C. , Jang , H., Zeng, W., "A new data-mining based approach for network intrusion detection," *Seventh Annual Communications Networks and Services Research Conference, SACNSRC*, 2009.

[4] Zhang, J., Mohammad, Z., "Anomaly based network intrusion detection with unsupervised outlier detection," *ICC'06. IEEE International Conference , ICC*, 2006.

[5] Buczak, L. A., Guven, E., "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *International Journal of Research, IJOR*, 2017.

[6] Kuang, F., Xu, W., Zhang, S., "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Applied Soft Computing* , ELSEVIER, pp. 179, 2014.

[7] Qu, F., Zhang, J., Shao , Z., "An Intrusion Detection Model Based on Deep Belief Networks," *Second International Conference on Advanced Cloud and Big Data*, IEEE, pp. 247, 2014.

[8] Kim, J., Kim, J., Thu et al, L. T. H., "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection," *International Conference on Platform Technology and Service*, IEEE, pp. 1-5 2016.

[9] Ding, Y., Zhai, Y., "Intrusion Detection System for NSL-KDD Dataset Using Convolutional Neural Networks," *The International Conference on Computer Science and Artificial Intelligence (CSAI)*, ACM, pp. 81, 2018.

A.2

Part of the thesis has been accepted in the 3rd IEEE International Conference on Telecommunications and Photonics (ICTP) 2019 which explores the voting classifiers in the results reported in this research work using Scikit learn.

Md. Raihan-Al-Masud and Hossen Asiful Mustafa. “Network Intrusion Detection System Using Voting Ensemble Machine Learning”, in *the 3rd IEEE International Conference on Telecommunications and Photonics (ICTP)*, December 2019.