A Hybrid Technique using Bagging, Boosting and Majority Voting Ensemble for Network Intrusion Detection

Project Report Submitted in Partial Fulfillment of the requirements for the Degree of

MASTER OF TECHNOLOGY
In
Computer Science and Engineering
By

Iram Abrar
(17083110007)

Under the Supervision of
Dr. Sajad Mohammed Khan,
Scientist B

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF KASHMIR
Hazratbal, 190006
2020

Certificate

This is to certify that this project report entitled ***"A Hybrid Technique using Bagging, Boosting and Majority Voting Ensemble for Network Intrusion Detection"*** by **Iram Abrar** (Roll No. 17083110007), submitted in partial fulfillment of the requirements for the M-tech Project in the department of Computer Science of the University of Kashmir, during the academic year 2020, is a bonafide record of work carried out under our guidance and supervision.

Signature
Prof. M. Arif Wani
HEAD OF THE DEPARTMENT

Signature
Dr. Sajad Mohammed Khan
SUPERVISOR

Declaration

I declare that I have written the Master's Thesis titled "**A Hybrid Technique using Bagging, Boosting and Majority Voting Ensemble for Network Intrusion Detection**" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis. As the author, I furthermore declare that, with respect to the creation of this Master's Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights.

Iram Abrar

Acknowledgement

I wish to express my deep sense of gratitude to my supervisor **Dr. Sajad Mohammed Khan**, Scientist B, Department of Computer Science, University of Kashmir for his consistent encouragement, expert advice, and incalculable guidance to carry out this project. I am thankful to him for giving me an opportunity to work under his supervision and for providing me with a great environment to carry out the current research with ease. It has a great honor to work under him.

I would also like to thank **Dr. Faheem Masoodi**, and **Prof. Arif Wani (HoD)**, Department of Computer Science, University of Kashmir for their valuable guidance and encouragement throughout my research work. I am grateful to all the faculty and staff members of Department of Computer Science, University of Kashmir who have always supported me during my Masters project.

Above all the credit of compulsion of this work goes to my parents. I express my humble gratitude towards my parents for their decades of support, whose patience, encouragement, unfailing love and blessings are reflected in each and every word of this dissertation.

Iram Abrar

Abstract

Network security has been one of the most important problems in computer networks. Intrusion is the most publicized threats to security. In recent years, intrusion detection has emerged as an important field for network security. It is an important technology in business sector as well as an active area of research. In information security, intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource. It plays a very important role in attack detection, security check and network inspect. One of the primary challenges to intrusion detection is the problem of misjudgment, misdetection and lack of real time response to the attack. In the recent years, as the second line of defense after firewall, the intrusion detection technique has got rapid development. Various data mining techniques such as clustering, classification and association rule discovery are being used for intrusion detection.

Recent advances in information technology have produced a wide variety of machine learning methods, which can be integrated into an IDS. Various machine learning techniques have been applied to improve the performance of intrusion detection systems, among them ensemble learning has received a growing interest and is considered as an effective method. In the current research work, different classifier like support vector machine, decision tree, random forest, K nearest neighbor were used in combination and three ensemble approaches, namely bagging, boosting and majority voting were used to obtain the final results. The performance of the model was evaluated on a well known NSL-KDD dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1 Introduction

Security has been one of the most important problems in Computer Networks and in recent years, intrusion detection has emerged as an important field for network security. A wide range of activity falls under this definition, including attempt to de-stabilize the network as a whole, gain unauthorized access to files or privileges, simply mishandling and misuse of software. Security measures can be adopted to stop all such attacks. The goal of intrusion detection [1] is to build a system which would automatically scan network activity and detect such intrusions. Once an attack is detected, the system administrator could be informed and thus take corrective action. Detecting such attacks is not only provides information on damage assessment, but also helps to prevent future attacks. These attacks are usually detected by tools referred to as intrusion detection system. Intrusion Detection System (IDS) obtain better results when each class of attacks is treated as a separate problem and handled by specialized algorithms.

Network IDS's analyze traffic to detect on-going and incoming attacks on a network. Nowadays, commercial IDS's mainly use a database of rules to detect attacks on a network or on a host computer. Intrusion detection systems are used to monitor devices which detect intrusions on a computer or a network. An intrusion detection system is an indispensable tool for network administrators because without such a device, it would be impossible to analyze the huge amount of packets traversing current networks every second. The variants of known attacks as well as new attacks can often go through the system without being detected.

An intrusion detection system (IDS) is software and/or hardware designed to detect unwanted attempts at accessing, manipulating or disabling of a computer system, mainly through a network, such as the internet. One of the main challenges in the security management of large-scale high-speed networks is the detection of anomalies in network traffic. A secure network must provide the following:

- **Data confidentiality:** Data that is being transferred through the network should be accessible only to the authorized users.
- **Data integrity:** Data integrity implies that data should not be modified during its transmission from source to destination that is no corruption or data loss is accepted either from random events or malicious activity.

- **Data availability:** The network should be resilient to Denial of Service attacks and data should be available to authentic users at all times.

1.1 Motivation

In the past two decades, networks have experienced tremendous growth that has speed up a shift in computing environments from centralized computer systems to network information systems. A large volume of valuable information such as personal profiles and credit card information is distributed and transferred through networks. Hence, network security has become more important than ever. However, given open and complex interconnected network systems, it is difficult to establish a secure networking environment. Intruders endanger system security by crashing services, changing critical data, and stealing important information. Intrusion detection systems (IDSs) are designed to discover malicious activities that attempt to compromise the confidentiality, integrity of computer systems. Unlike a firewall that filters bad traffic, IDS analyzes packets to detect malicious attack attempts. Based on the report of the CSI/FBI computer crime and security survey in 2006, the IDS has become the fifth most widely used security technology. Undoubtedly, intrusion detection system has become critical component in network security. Therefore, two factors need to be considered to ensure IDS effectively. First, the IDS should deliver reliable detection results. The detection method should be effective in discovering intrusions since poor detection performance ruins the trustworthiness of the IDS. Secondly, the IDS should be able to survive in hostile environments. However, it is challenging for IDSs to maintain high detection accuracy. An IDS [3] that uses attack signatures to detect intrusions cannot discover novel attacks. As the number of new intrusions increases, these IDSs are becoming incapable of protecting computers and applications. Therefore, a detection approach that is able to discover new attacks is necessary for building reliable IDSs. Machine learning- based detection methods provide insights for identifying novel attacks. Machine learning is the ability of a machine that automatically improves its performance through learning from experience. Machine learning techniques are used to study normal computer activities and identify anomalous behaviors that deviate from the normal as intrusions. Although these anomaly-based IDSs are able to detect novel attacks, most of them suffer from a

high error rate due to a deficiency in their discrimination ability. Since these detection approaches can only distinguish anomalous attacks, they miss many attacks that are not significantly different from normal behaviour.

A computer system should provide confidentiality, integrity and assurance against denial of service. However, due to increased load and connectivity more and more system is subject to attack by intruders. These attempts try to exploit flaws in the operating system as well as in the application programs. In fact, it is not possible to build a completely secure system. It can have cryptographic methods but they have their own problems as passwords can easily be cracked, users can lose their passwords and entire crypto-system can be broken. Even a truly secure system is vulnerable to abuse by insiders who misuse their privileges. Also, we need a balance between access control and user efficiency. So, we need a system which is real time i.e. we would like to detect intrusion as soon as possible and take appropriate action. Thus, IDS is reactive rather than proactive.

1.2 Intrusion Detection System

An intrusion in an information system is an activity that violates the security policy of the system, i.e, it is a deliberate unauthorized attempt to:

- Access information
- Manipulate information
- Render system unreliable.

Intrusion detection is a process which is used to identify the intrusion, and is based on the belief that the intruder behavior will be significantly different from the legitimate user. Intrusion Detection System (IDS) [4] are usually deployed along with other preventive security mechanisms, such as access control and authentication, as a second line of defense that protects information systems. There are several reasons that make intrusion detection a necessary part of the entire defense system. Firstly, many traditional systems and applications were developed without security in mind. In other cases, systems and applications were developed to work in a different environment and may become vulnerable when developed in the current environment. Intruders can be classified as:

- **External penetrators:** These are unauthorized users of the system.

- **Internal penetrators:** These are legitimate users of the system who access computer resources for which they are not authorized. These can be divided as: masqueraders who either steal an identity of other users or pretend to be them.

An intrusion can be defined as "any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource". Intrusion detection system (IDS) can detect and identify intrusion behavior or intrusion attempts in a computer system by monitoring and analyzing network packet or system audit log, and then send intrusion alerts to system administrators in real time. To apply data mining techniques in intrusion detection, first, the collected data needs to be preprocessed and then it is converted to a format suitable for mining processing. Next, the reformatted data is used to develop a classification model. So, in this report a supervised learning classifier is used for intrusion detection to detect unwanted attempts at accessing, manipulating, and disabling of computer system, mainly through a network. The goal of IDS is to detect malicious traffic.

Intrusion Detection System is proposed to improve computer security because it is not feasible to build completely secure systems [2]. In particular, IDSs are used to identify, assess, and report unauthorized or unapproved network activities so that appropriate actions may be taken to prevent any future damage. Based on the information sources that they use, IDSs can be categorized into two classes: network-based and host-based. Network intrusion detection systems (NIDSs) analyze network packets captured from a network segment, while host- based intrusion detection systems (HIDSs) such as IDES (Intrusion Detection Expert System) examine audit trails or system calls generated by individual hosts.

NIDSs use software programs called sensors to collect network packets. Because raw packets cannot be used directly for detection, many sensors have preprocessing units that transform the packets into a useful format. As the volume of network traffic increases, many NIDSs employ multiple sensors and distributed computing to improve their processing capability. NIDSs can also detect IP-based attacks such as denial- of-service attacks which involve multiple computers. A host based IDS has difficulty in detecting these attacks since it monitors only information gathered from the computer system. NIDS is gaining popularity since more and more systems are connecting over networks.

IDSs can also be categorized according to the detection approaches they use. Basically, there are two detection methods: misuse detection and anomaly detection. The major deference between the two methods is that misuse detection identifies intrusions based on features of known attacks while anomaly detection analyzes the properties of normal behavior. IDSs that employ both detection methods are called hybrid detection-based IDSs.

1.2.1 Misuse Detection

Misuse detection catches intrusion in terms of the characteristics of known attacks. Any action that conforms to the pattern of a known attack or vulnerability is considered as intrusion. The main issues in misuse detection system are how to write a signature that encompasses all possible variations of the pertinent attack and how to write signatures that do not match non-intrusive activity. Block diagram for misuse based intrusion detection system has been depicted below.

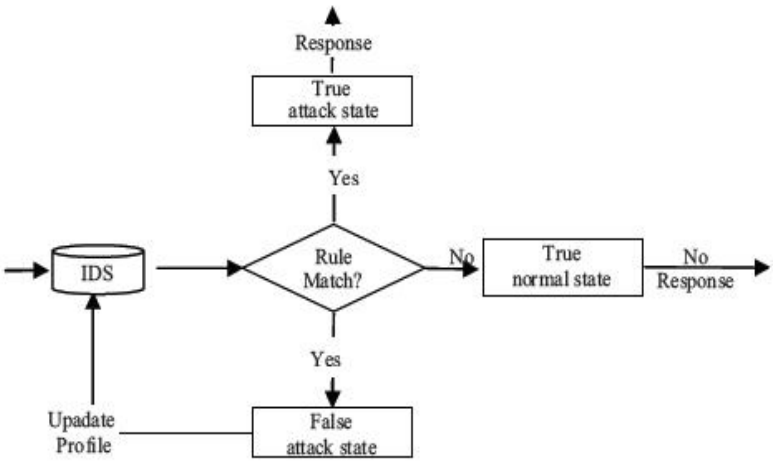

Figure 1: Misuse detection

Misuse detection identifies intrusions by matching monitored events to patterns or signatures of attacks. The attack signatures are the characteristics, associated with successful known attacks. The major advantage of misuse detection is that the method possesses high accuracy in detecting known attacks. However, its detection ability is limited by the signature database. Unless new attacks are transformed into signatures and

added to the database, misuse-based IDS cannot detect any attack of this type. Different techniques such as expert systems, signature analysis, and state transition analysis are utilized in misuse detection.

An expert system-based IDS generates a set of rules that describes known attacks. Then, the audit data that the IDS monitor is translated into facts accompanied by their semantic signification. An inference engine applies the rule set to look for any matches with the facts. Because the method makes use of semantic information in drawing its conclusion, it increases the abstraction level of the audit events. Signature analysis follows the same approach as the expert system method, but it exploits the patterns of attacks in a different way. State transition analysis uses a graphical notation to present the process of intrusions as a series of state changes from an initial secure state to a target compromised state. The state-transition analysis technique is applied to different types of IDSs and has many implementations.

1.2.2 Anomaly Detection

It is based on the normal behavior of a subject (e.g. a user or a system). Any action that significantly deviates from the normal behavior is considered as intrusion. There is a important difference between anomaly based and misuse based technique that the anomaly based tries to detect the compliment of bad behavior and misuse based detection system tries to recognize the known bad behavior. In this case we have two possibilities:

- **False positive**: Anomalous activities that are not intrusive but are flagged as intrusive.
- **False Negative**: Anomalous activities that are intrusive but are flagged as non intrusive.

The block diagram of anomaly detection system has been depicted below.

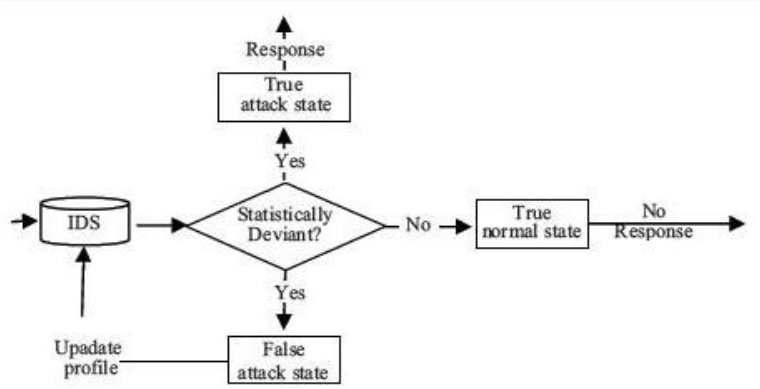

Figure 2: Anomaly detection

Anomaly detection assumes that intrusions are anomalies that necessarily differ from normal behavior. Basically, anomaly detection establishes a profile for normal operation and marks the activities that deviate significantly from the profile as attacks. The main advantage of anomaly detection is that it can detect unknown attacks. However, this advantage is paid for in terms of a high false positive rate because, in practice, anomalies are not necessarily intrusive. Moreover, anomaly detection cannot detect the attacks that do not obviously deviate from normal activities.

As the number of new attacks increases rapidly, it is hard for a misuse detection approach to maintain a high detection rate. In addition, modeling attacks is a highly qualified and time consuming job that leads to a heavy workload of maintaining the signature database. On the other hand, anomaly detection methods that discover the intrusions through heuristic learning are relatively easy to maintain.

Chapter 2

Literature Survey

2 Literature Survey

A number of research works that have been conducted in the field of IDS were investigated and this section presents an overview of the same. Recently, machine learning algorithms such as support vector machine, decision trees, multilayer Perceptron, K-Nearest neighbor, random forest has been employed to classify the data as normal or intrusive. Although single classifier are extensively used for intrusion detection, recent research works have revealed that ensemble based approaches yield a better performance due to which ensemble has received a wide attention in the recent times. As ensemble is composed of multiple classifiers, they have strong generalization ability and thus, are able to predict better results compared to single classifier where a single hypothesis is used for classification.

In order to enhance the performance of an IDS, Aburomman et al. [1] proposed an ensemble based technique in which SVM and KNN were used as base classifiers and particle swarm optimization was used for assigning weights to the classifiers and the final result was obtained on the basis of majority voting. A comparative study of three ensemble techniques namely, PSO based ensemble, LUS (local unimodal sampling) improved PSO based model and weighted majority algorithm based ensemble was carried out using six different datasets obtained from KDD-99 dataset. On the basis of the results, it was concluded that the LUS improved PSO (where LUS was used as meta-optimizer) outperformed the other classifier with an average accuracy of 92.7884 percent.

Bamakan et al. [2] proposed an efficient intrusion detection system using NSL-KDD dataset. A modified version of particle swarm optimization namely time varying chaotic particle swarm optimization (TVCPSO) was used in the proposed method for feature selection in order to increase the detection rate and decrease the false alarm rate. The performance of the system was evaluated using two classifiers namely support vector machine and multiple criteria linear programming (MCLP). An accuracy of 96.88% was obtained in case of TVCPSO-MCLP and an accuracy of 97.84 was obtained in case of TVCPSO-SVM. Further the detection rate of 97.23% and 97.03% and false alarm of 2.41% and 0.87% was attained for TVCPSO-MCLP and TVCPSO-SVM respectively.

Jabber et al. [3] proposed an ensemble based approach using random forest and average one-dependence estimator (AODE) for intrusion detection. The efficiency of the

proposed model was tested on Kyoto dataset in which out of 24 features only 15 features were taken into consideration. During the pre-processing phase, numeric attributes were converted to binary and then the dataset was trained and tested using ensemble approach which was built using RF and AODE. The efficiency of the ensemble based model was 90.51% percent whereas using these two algorithms individually an accuracy of 89.68% and 89.34% was obtained for AODE and RF respectively using WEKA tool. Furthermore, the performance of the proposed system was compared with the existing algorithms such as Naïve Bayes, J48 and PART based on which the authors claimed that the accuracy of the proposed ensemble approach outperforms that of the individual algorithms.

Timcenko et al. [4] proposed an ensemble based approach using different machine learning algorithms and the performance of the model was evaluated on UNSW-15 dataset. A number of ensemble techniques namely, Bagged trees, AdaBoost, GentleBoost, LogitBoost and RUSBoost were used where decision tree was selected as base classifier. Based on their results, they claimed that the Bagged tree and GentleBoost algorithms outperformed the other classifiers with an average detection rate of 95.6% and 95.45% respectively for denial of service, fuzzers, reconnaissance and exploits attacks. Similarly, their experimental results revealed that RUSBoost was less efficient having an average detection rate of 86.15% for DoS, Fuzzers, Reconnaissance and exploits attacks.

Ma et al. [5] proposed an ensemble approach in which spectral clustering and deep neural network were combined to form an IDS and the performance of the system was evaluated on two well known datasets namely, KDD-99 and NSL-KDD where the datasets were divided into six subsets and then spectral clustering algorithm was employed to cluster the similar data. The clustered data was then fed to deep neural network on the basis of which results were obtained. From the results, it was concluded that the proposed model SCDNN outperformed other algorithms such as back propagation neural network, support vector machine, random forest and Bayes tree. Thus, according to [5] the model could be used efficiently to detect intrusion on a large networks and works well for sparse attack types like U2L and R2L.

Gautam et al. [6] used three experimental setups to perform a comparative study on KDD-99 dataset where naïve bayes, partial decision tree (PART) and adaptive boost

classifiers were used to detect intrusion. In the first case, all 41 features present in the dataset were taken into consideration based on which performance of the system was evaluated. The experimental results revealed that the PART algorithm outperformed the other two classifiers with an accuracy of 99.9601%. In the second case, significant features present in the dataset were selected using filter based information gain technique based on which it was concluded that the results for PART (99.9589%) and adaptive boost (97.9073%) were better as compared to that of naïve bayes (91.9818%). In the third case an ensemble based bagging method was used to analyze the performance of the system and the final result was obtained on the basis of majority voting. An accuracy of 99.9732% was achieved using the ensemble approach which outperformed the other three classifier used for intrusion detection.

In order to increase the accuracy of an IDS, Sahu et al. [7] proposed an ensemble of K means clustering and support vector machine where former is used to divide the data into two clusters and the latter is used to identify the labels based on which the data is classified as normal or intrusive. The performance of the proposed system was evaluated on three dataset namely, NSLKDD, GureKDDCup and KDDCorrected dataset and an accuracy of 99.7%, 99.5% and 100% was achieved respectively.

Chen et al. [8] proposed a new clustering based ensemble approach for intrusion detection where NSL KDD dataset was used to evaluate the performance of the system. The dataset was divided into four subsets and each one of them was trained using a different algorithm namely, DBSCAN, one-SVM, agglomerative clustering and expectation maximization and the final result was obtained on the basis of majority voting. A comparative study of this model with other existing models was carried out based on which it was concluded that the proposed model outperforms them in terms of detection rate (91.0333%) and false alarm rate (2.2633%).

In order to increase the accuracy of an intrusion detection, Rajasekaran et al. [9] proposed an ensemble approach using multiclass support vector machine and KNN where incremental particle swarm optimization (IPSO) was used to optimize the accuracy of classification. In the proposed system, intelligent agent based attribute selection algorithm and ICRFFSA was used for feature selection and the performance of the system was evaluated on KDD-99 dataset. A comparative study with the existing single,

hybrid and ensemble models was carried out based on which they concluded that their model had achieved better results. A detection rate of above 96% was achieved for DoS, U2R, probe attack where as in case R2L a detection rate of around 85% was achieved.

Adeniran et al. [10] proposed a stacking based intrusion detection system. The performance of the system was evaluated on KDD-99 dataset where Naïve Bayes, multilayer perceptron were used as base classifier. In the proposed system, J48 was used as a meta-classifier and filter based gain ratio was used for feature selection. The performance of the ensemble model was compared with that of the individual models on the basis of which it was concluded that the model outperformed the existing single classifier in terms of accuracy and false alarm rate. The accuracy of this system was above 99% for DoS, probe and R2L and above 75% for U2R.

Salo et al. [11] proposed an ensemble based approach in which information gain and principle component analysis were used to select the significant features based on which the model was trained/tested. The performance of the system was evaluated on three well known datasets ISCX-2012, NSL-KDD and Kyoto 2006+ dataset using SVM, instance based learning algorithm and multilayer perceptron classifier. The experimental results revealed that the proposed system outperformed the individual classifier with an accuracy of 99.01% on ISCX, 98.95% on Kyoto dataset and 98.24% on NSL-KDD dataset.

Pham et al. [12] proposed a tree based ensemble model where bagging and boosting based ensemble approaches were employed to improve the effectiveness of an IDS. Two feature selection methods were employed based on which 25 and 35 features were selected from the NSL-KDD dataset. Among the two the former one was the combination of Naïve Bayes and leave one out and the latter one used filter based gain ratio to select the significant features from the dataset. J48, random forest (RF) and REPtree were used as base classifiers to classify the data as normal or intrusive. Using 25 features an accuracy of 83.22% was obtained in case of bagged REPtree whereas an accuracy of 84.25% was achieved in case of J48 on 35 features.

In order to increase the effectiveness of Intrusion detection system, an ensemble based approach was proposed by Zhou et al. [13] where correlation based bat algorithm was used for feature selection. The performance of the system was analyzed on three well known datasets KDD-99, NSL-KDD and CICIDS-2017 where C4.5, random forest and

forest penalizing attribute classifiers were used and the final result was obtained through voting classifier. The accuracy of the model was 97.56% for KDD-99, 99.14% for NSL-KDD and 96.76% for CIC-IDS 2017 dataset.

Wang et al. [14] proposed an ensemble of SVM, KNN, Naïve bayes, classification and regression tree (CART) and random forest for categorizing the applications as malware or benign. Based on static analysis, 34630 features were extracted from the dataset on the basis of which the model was trained and tested. The performance of the system was evaluated on application dataset comprising of 107327 benign applications and 8701 malware applications and an accuracy of 99.39% for malware applications and 82.93% for benign applications was achieved.

Tama et al. [15] proposed an ensemble based approach where particle swarm optimization and filter based correlation method were used in combination for feature selection. The performance of the system was analyzed on NSL-KDD dataset where CART, random forest and C4.5 classifiers were used to evaluate the performance of the system. In the proposed model when the number of particles in PSO was set to fifty, an accuracy of 99.805% was achieved, outperforming the individual classifiers.

Kanakarajan et al. [16] proposed a tree based ensemble approach namely, GAR forest in which metaheuristic GRASP with annealed randomness was used to increase the diversity of the model. Information gain, symmetrical uncertainty and correlation based feature subsets were used for feature selection and the performance of the system was evaluated on NSL-KDD dataset. A comparative study of GAR forest with other existing algorithm namely, random forest, C4.5, naïve bayes and MLP was carried out based on which it was concluded that GAR forest outperformed the other algorithms with an accuracy of 85.0559% for binary class (32 features were used) and 78.9035% for multiclass (when 10 features were used).

Haq et al. [17] proposed an ensemble approach in which a wrapper based feature selection methods viz: best first, genetic search and rank search method were used to select twelve significant features from the NSL-KDD data set based on which the model was trained and tested. For classification purpose, three algorithms namely Bayesian network, naïve bayes and J4.8 were used in combination and on the basis of majority voting final result was obtained. According to them [17] the proposed model

outperformed the individual classifiers in terms of true positive rate (98 %) and false positive rate (0.021%).

Osanaiye et al. [18] proposed an ensemble method which was particularly designed to detect denial of service attack on cloud where multi-feature filter selection method (EMMFS) was used to select significant features present in the dataset. In EMMFS, four filter based approaches namely, information gain, gain ratio, chi-squared statistics and Relief F were used based on which 13 features were selected and the performance of the system was evaluated on NSL-KDD dataset using WEKA tool. In the proposed system, decision tree algorithm was used for classifying the data as normal or attack and an accuracy of 99.67%, detection rate of 99.76% and false alarm rate of 0.42% percent was achieved.

Table 1: Performance comparison of bagging based ensemble techniques

Reference	**Classifiers/ technique**	**Dataset**	**Results**	**Year of Publication**
Aburomman et al. [1]	SVM, KNN and PSO	KDD-99	92.7884%	2016
Bamakan et al.[2]	TVCPSO- MCLP	NSL-KDD	96.88%	2016
	TVCPSO- SVM	NSL-KDD	97.84%	
Jabber et al.[3]	RF, AODE	Kyoto 2006	90.51%	2017
Timcenko et al. [4]	Bagged tree	UNSW-15	95.6%	2017
	Gentleboost	UNSW-15	95.45%	
	Adaboost	UNSW-15	93.65%	
	Logitboost	UNSW-15	94.75%	
	Rusboost	UNSW-15	86.15%	
Ma et al.[5]	Spectral clustering and deep neural network	KDD-99 and NSL-KDD	NA	2016

Gautam et al. [6]	Naïve Bayes, PART and Adaboost	KDD-99	99.9732%	2018
Sahu et al. [7]	K-means and SVM	NSL-KDD	99.7%	2019
	K-means and SVM	Gure-KDD	99.5%	
	K-means and SVM	KDD-corrected	100%	
Chen et al.[8]	DBSCAN, One SVM, agglomerative clustering and expectation maximization	NSL-KDD	91.0333%	2017
Rajasekaran et al.[9]	Multiclass SVM, KNN and IPSO	KDD-99	96%(DoS,U2R Probe)	2017
	Multiclass SVM, KNN and IPSO	KDD-99	85%(R2L)	
Adeniran et al.[10]	Naïve bayes, MLP and J48	KDD-99	99%(DoS Probe,R2L)	2017
	Naïve bayes, MLP and J48	KDD-99	75%(U2R)	
Salo et al. [11]	SVM, instance based-learning, and MLP	ISCX-2012,	99.01%	2019
	SVM, instance based learning and MLP	NSL-KDD	98.24%	
	SVM, instance based learning and MLP	Kyoto 2006	98.95%	
Pham et	J48, random forest	NSL-KDD	84.25%	2018

al.[12]	and REPtree			
Zhou et al. [13]	C4.5, RF, and forest penalizing attributes	KDD-99	97.56%	2015
	C4.5, RF, and forest penalizing attributes	NSL-KDD	99.14%	
	C4.5, RF, and forest penalizing attributes	CICIDS-17	96.76%	
Wang et al. [14]	SVM, KNN, Naïve Bayes, CART and RF	Application dataset	99.39% for Malware applications	2018
	SVM, KNN, Naïve Bayes, CART and RF	Application dataset	82.93% for Benign applications	
Tama et al.[15]	CART,RF and C4.5	NSL-KDD	99.805%	2015
Kanakarajan et al.[16]	GAR forest	NSL-KDD	85.0559% (Binary class) 78.9035% (multiclass)	2015
Haq et al.[17]	Bayesian network, Naïve bayes and J4.8	NSL-KDD	98 % TP and 0.021% FP	2015
Osanaiye et al. [18]	Information gain, gain ratio, chi-square statistics, and Relief F	NSL-KDD	99.67%	2016

Chapter 3

Dataset and Algorithm

3 Data set and classifiers

To evaluate the performance of an intrusion detection system, an adequate dataset must be available so that the performance of the system can be analyzed before deploying it in the real- world. The dataset which is used to analyze the performance of the system must be complete and comprehensive. Similarly, various classifiers which have been employed for network intrusion detection have been briefly discussed in the current section.

3.1 NSL-KDD dataset

The inherent drawbacks in the KDD cup 99 dataset [3] has been revealed by various statistical analyses has affected the detection accuracy of many IDS modeled by researchers. NSL-KDD dataset [5] is a refined version of its predecessor. It contains essential records of the complete KDD data set. This data set is publicly available for researchers through the website and has the following advantages over the original KDD data set:

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There are no duplicate records in the proposed test sets; therefore, the performance of the learners is not biased by the methods which have better detection rates on the frequent records.
- The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
- The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

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 last attribute contains data

about the various 5 classes of network connection vectors and they are categorized as one normal class and four attack class. The 4 attack classes are further grouped as DOS, Probe, R2L and U2R.The description of the attack classes is given below.

Table 2: Mapping of attack class with attack type

Attack Class	**Attack Type**
DOS	Back, Land, Neptune, Pod, Smurf,Teardrop,Apache2, Udpstorm, Processtable, Worm
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

The attack classes present in the NSL-KDD data set are grouped into four categories [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. Relevant features to detect this type of attack are source bytes and percentage of packets with errors.
- Probing: Surveillance and other probing attack's objective is to gain information about the remote victim e.g. port scanning. Relevant features include duration of connection and source bytes.
- 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. Relevant features include number of file creations and number of shell prompts invoked.
- 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

.Network level features are used to detect this attack which includes duration of connection and service requested and host level features number of failed login attempts.

3.2 Classifiers

An effective IDS has a reasonable detection rate, less false alarm rate, and less training time; however, a traditional IDS lacks most of these characteristics. Since intrusion detection is primarily a classification problem that identifies data as normal or intrusive, ML algorithms have been used to develop active IDS. They have a proficient generalization ability, which yields excellent performance. Conversely, the set-up and execution of such systems can be complicated and requires an immense amount of data that can be trained/ tested in the system, making the process even more complicated. Despite these facts, it is necessary to have an effective IDS where the security of the system is not held back, so to reduce computational complexity, feature selection methods have been employed [19, 20]. Various ML algorithms used in recent times for network intrusion detection, like support vector machine (SVM), decision tree (DT), and artificial neural networks (ANN), have been discussed below.

3.2.1 Support Vector Machine (SVM)

SVMs were initially proposed by Vapnik (1995) for solving classification problems and regression analysis. SVM is a supervised learning technique that is trained to classify different categories of data from various disciplines. These have been used for two-class classification problems and are applicable on both linear and non-linear data classification tasks. SVM creates a hyperplane or multiple hyperplanes in a high-dimensional space, and the best hyperplane in them is the one that optimally divides data into different classes with the largest separation between the classes. A non-linear classifier uses various kernel functions to estimate the margins. The main objective of these kernel functions (i.e., linear, polynomial, radial basis, and sigmoid) is to maximize margins between the hyper-planes. Recently, many highly promising applications have been developed by researchers because of the increasing interest in SVM. They have been widely used in image processing and pattern recognition applications [21].

The below figure illustrates the architecture of the SVM classification model in the proposed intrusion detection system. The kernel function uses squared Euclidean distance between two numeric vectors and maps input data to a high dimensional space to optimally separate the given data into their respective attack classes. In addition to performing linear classification, SVM can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

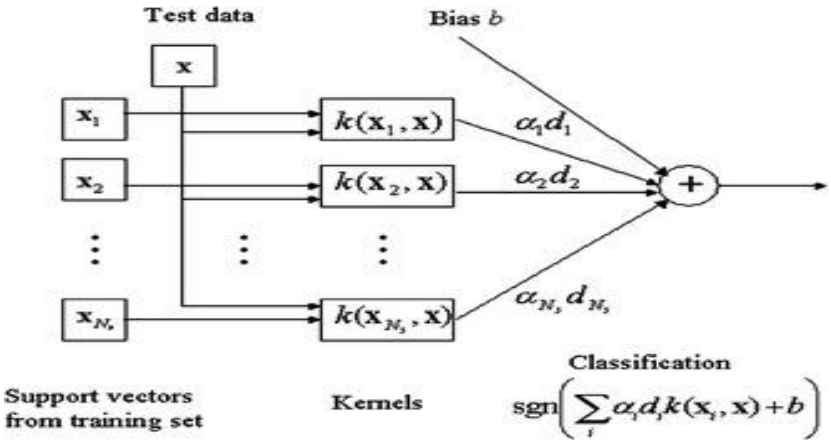

Figure 3: Support vector machine

When data is unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data. The support-vector machine algorithm, applies the statistics of support vectors, developed in the support vector machine algorithm, to categorize unlabeled data, and is one of the most widely used classification algorithms in industrial applications.

3.2.2 Random forest (RF)

Random forest (RF) is an ensemble classifiers, which is used for classification and regression analysis on the intrusion detection data. RF works by creating various decision trees in the training phase and output class labels those have the majority vote. RF attains high classification accuracy and can handle outliers and noise in the data. RF is used in

this work because it is less susceptible to over-fitting and it has previously shown good classification results [22].

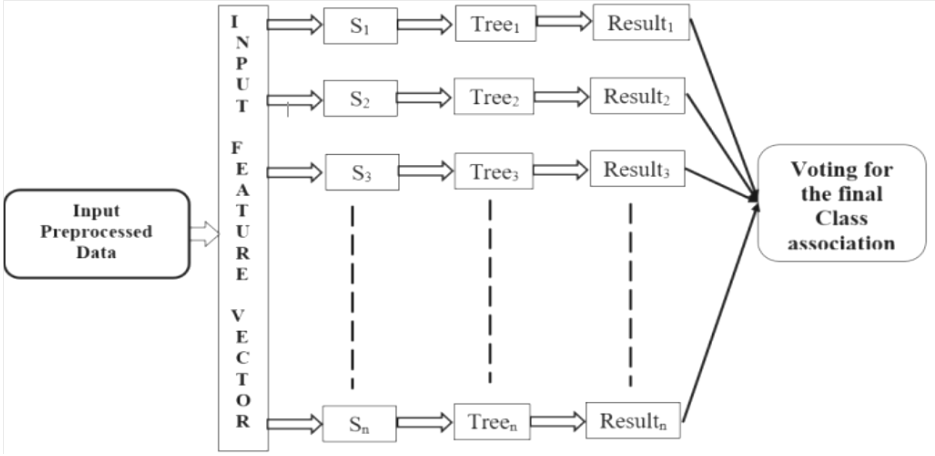

Figure 4: Architecture of the Random forest for intrusion detection system

Above figure shows the implementation of the random forest classification model in the data classification in the proposed system. A pre-processed sample of n samples is fed to the random forest classifier. RF creates n different trees by using a number of feature subsets. Each tree produces a classification result, and the result of the classification model depends on the majority voting. The sample is assigned to the class that obtains highest voting scores. The previously attained classification results indicate that RF is reasonably suitable in the classification of such data because in some cases, it has obtained better results than have other classifiers. Other advantages of the RF include its higher accuracy than Adaboost and fewer chances of over-fitting.

3.2.2 K-nearest Neighbor (KNN)

KNN classification algorithm is a data mining algorithm which is based on the idea that in a sample space, if most of its *K* nearest neighbor samples belong to a category, then the sample belongs to the same category. The nearest neighbor refers to the single or multidimensional feature vector that is used to describe the sample on the closest, and the closest criteria can be the Euclidean distance of the feature vector. The data points that have minimum distance between them are placed in one cluster as they have similar characteristics. The procedure is repeated until the data has been placed in a cluster [23]. KNN is easy to implement and works well for large datasets.

3.2.3 Extra Trees Classifier (ETC)

It is a type of ensemble learning technique which aggregates the results of multiple de correlated decision trees collected in a forest to output it's classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. Here each decision stump will be built with the following criteria:

- All the data available in the training set is used to build each stump.
- To form the root node or any node, the best split is determined by searching in a subset of randomly selected features of size squared by number of features. The split of each selected feature is chosen at random.
- The maximum depth of the decision stump is one.

In extra trees classifier, the features and splits are selected at random; hence, extremely randomized tree. Since splits are chosen at random for each feature in the extra trees classifier, it's less computationally expensive than a random forest.

3.2.4 Multi-layer Perceptron (MLP)

MLP neural networks consist of units arranged in layers. Each layer is composed of nodes and is fully connected to every node in subsequent layers. Each MLP is composed of a minimum of three layers consisting of an input layer, one or more hidden layers and an output layer. The input layer distributes the inputs to subsequent layers. Input nodes have liner activation functions and no thresholds. Each hidden unit node and each output node have thresholds associated with them in addition to the weights. The hidden unit nodes have nonlinear activation functions and the outputs have linear activation functions. Hence, each signal feeding into a node in a subsequent layer has the original input which is passed through an activation function that may or may not be linear.

3.2.5 Particle swarm optimization (PSO)

It is an evolutionary, population-based, stochastic algorithm presented by Kennedy and Eberhart [24] and is used to obtain the best solution to a given problem. It is based on social behavior of group of birds where each bird adjusts its position depending upon the

position of its neighbor to obtain the food. Similarly, in this algorithm, each particle denotes a possible solution to the problem and the information is exchanged among the particles based on which they adjust their position. Initially all the particles are initialized with a random value and on the basis of the fitness function their position are updated iteratively so that the best solution is attained and the resources are utilized in an efficient manner [25]. PSO is used to select features in intrusion detection where the chosen features are denoted by 1 and the other attributes which are not selected are represented by 0. The chosen features are then forwarded to the classifier which computes the fittest of particle based on which the detection rate is obtained.

3.3 Experimental setup

The experiments have been conducted by a decent powered Intel® core ™ i7-7500U CPU @ 2.70GHz (4 CPU's) with a memory size of 8GB. Python which is an open source scripting language was used in the current research work. Since it is open source, it is freely available to download. Most GNU/Linux distributions already have python pre-installed. Since python is written in C, it is easily portable to other system (and so would this distributed firewall). The python language has a clear syntax, is object oriented (everything in python is actually an object), and has numerous bindings to other language/libraries. When a lot of work needs to be done, for example implementing a distributed firewall, Perl code tends to become ugly pretty fast. Python is easier because of it strict syntax and object oriented nature. Python has been used as the development language with development environment being provided by Linux. Anaconda tools was used for providing integrated development environment.

Chapter 4

Ensemble

4 Ensemble

Ensemble technique can be employed to detect intrusion in the system where the performance of the system can be improved using different classifiers. It combines several weak classifiers to attain a powerful classifier that can be employed for the classification of data [26]. The classifier obtained is robust, and has a better prediction rate and good generalization ability, i.e., it can be used effectively for the classification of the unlabelled data. As it combines the merits of different algorithms, better results can be obtained using the ensemble approach, thereby reducing the error rate. Furthermore, if a classifier fails to identify an attack, the other classifier can detection the same, which significantly enhances the detection rate [27]. The ensemble is broadly categorized into homogeneous and heterogeneous [28]. In a homogeneous ensemble, same classifiers are used to train and test the data, whereas, in a heterogeneous approach, diverse classifiers such as SVM, MLP, and RF are employed for analyzing the system performance. Bagging and boosting are commonly used techniques for homogenous ensemble, while stacking and voting are used for a heterogeneous ensemble.

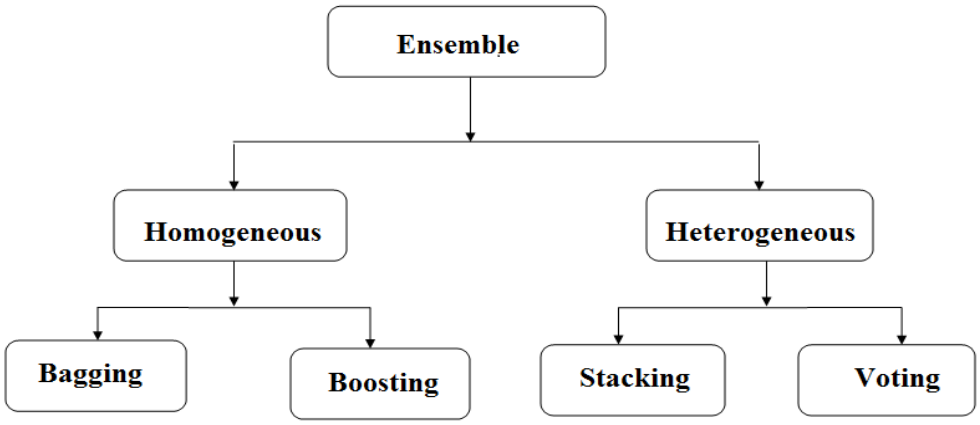

Figure 5: Hierarchical representation of various ensemble techniques

4.1 Bagging

The concept of bagging was given by Leo Breiman [29] to minimize the variance of the predictors. In bagging or bootstrap aggregation, D samples are selected randomly (with replacement) from the original dataset so that diverse training subsets, namely d_1, d_2,

$d_3 \ldots d_n$ are obtained. Every sample is equally likely to appear in the data subset due to the random sampling. These multiple datasets are also known as bootstrap samples, and different classifiers C_1, C_2, $C_3 \ldots C_n$ are trained using them. The output of these classifiers is then combined together to produce a final classifier, C*, which is more powerful as compared to individual weak classifiers. The final classifier has less error rate and a high prediction rate. The sample is classified by giving this sample as an input to the weak classifiers (C_1, C_2, $C_3 \ldots C_n$) and the results of individual weak base classifiers are joined to obtain a final result using the strong classifier, C* [30, 31], and hence the model performance is analyzed. Generally, voting or averaging is used to merge the results from the individual classifiers. Figure 6 depicts the general bagging process.

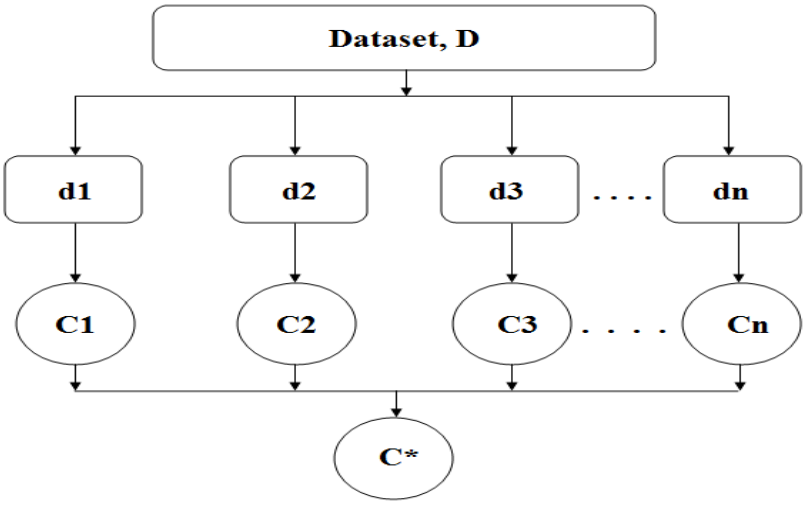

Figure 6: Systematic representation of bagging method

The performance of the system is enhanced using bagging as it reduces the variance [32], unstable classification [33], and over-fitting of data. It is a widely used technique for improving the accuracy of IDS as it yields high classification accuracy, low false-positive rate, and the model can be built in less amount of time using bagging as it is parallel in nature. In recent research works [34], a number of weak classifiers have been combined together in order to build a strong classifier using this technique, thereby improving the accuracy of IDS. Each instance present in the dataset is equally likely to appear in the training subset as the same weights are assigned to every instance. There exist a variety

of bagging, namely waging in which weights are associated with the instances which determine the probability of instance being chosen in the training subset [35].

4.2 Boosting

Boosting was presented by Schapire in 1990 [36]. It is a sequential, iterative ensemble technique that is used to exploit the dependencies between various classifiers for the proper classification of data [37]. In boosting the training dataset, D, consists of several instances, from which numerous subsets, namely d_1, d_2, d_3….d_n, are obtained. Initially, the same weights are allocated to all the instances so that they are equally probable of getting selected in the data subset, d_1. This dataset d_1 is then used for training the classification model M_1 and then the instances present in the actual dataset D are used for testing the model. Some of the instances present in the dataset might be misclassified in the model M_1, for example, there might be an instance which should belong to one class, but M_1 classifies it in another. The weights are updated, so that the previously misclassified instances are selected in the dataset d_2, which is obtained in the same way as d_1, and the model M_2 is employed for its training. This process is repeated till dataset d_n is obtained, which is used to train the model M_n. The weak classifiers M_1, M_2….M_n are then combined to form a strong classification model M*. An instance is then selected from the test dataset, which is fed to all models M_1, M_2….M_n for its classification. The output of all models obtained from M* is used to generate the result using voting [38]. Figure 7 depicts the general boosting process.

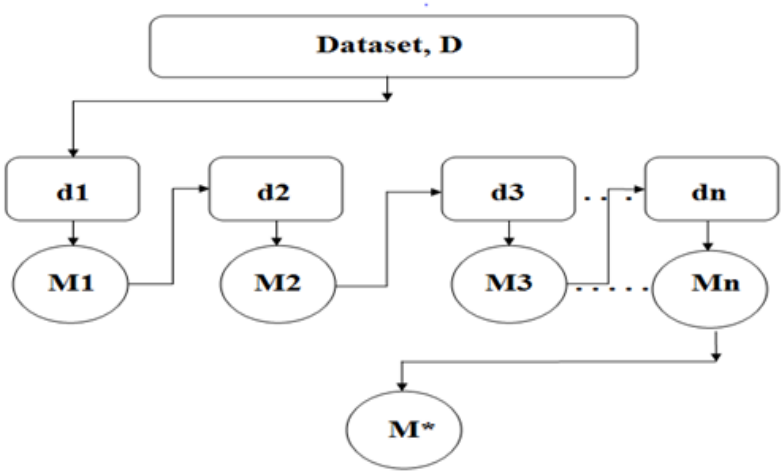

Figure 7: Systematic representation of boosting method

This method boosts the accuracy of IDS as it correctly classifies the misclassified samples to a large extent, and better ensembles are obtained using this approach as compared to that of bagging. However, the major limitations of this technique include over-fitting [39] and susceptibility to noise and outliers, particularly in the case of small datasets [40].

4.3 Majority voting

In majority voting dataset, D, is used to train n classifiers C_1, C_2, C_3….C_n. The performance of these classifiers is evaluated on a small test subset obtained from the original dataset, where every classifier predicts the class to which the test sample should belong. The final decision of allocating a label to the sample depends on the maximum number of votes received in favor of a particular class obtained from different classifiers. Furthermore, in the weighted majority voting, weights are assigned with the base classifiers, whereas in un-weighted majority voting, each base classifier has an equal weight assigned to it [41]. Majority voting is used to merge the results attained from individual weak classifiers in IDS so that the accuracy of the model can be enhanced in terms of detection rate. Moreover, prior studies on IDS [42] have revealed that voting has the capability of improving the performance of an IDS considerably, and it generally

outperforms other ensemble techniques. Figure 8 depicts the process of voting ensemble technique.

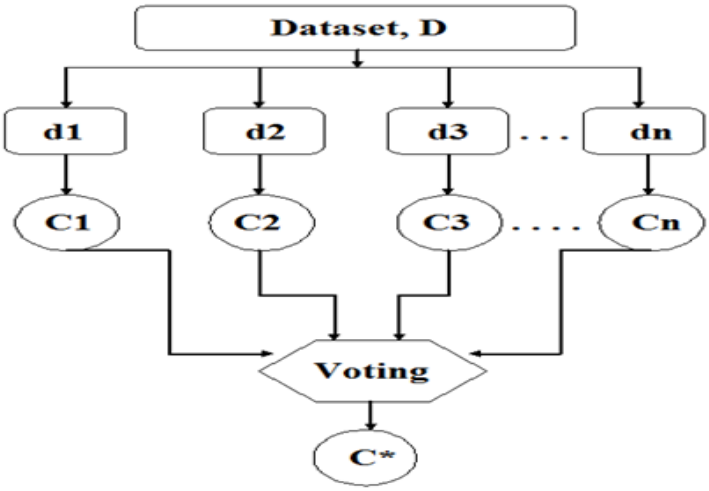

Figure 8: Systematic diagram of voting method

Chapter 5

Results and Discussion

5 Methodology

In the current research work, ensemble approach was used to combine the results from the several classifiers so that the efficiency of the system could be improved by increasing the detection rate and decreasing the false alarm rate. Decision tree was used for feature selection so that the training time and the computational complexity of the data could be reduced. The data was classified as normal or intrusive using one is to many approach and the performance of the system was optimized using particle swarm optimization technique. In order to evaluate the performance of the model, well known NSL-KDD data set was used. The steps used in the model are summarized and are precisely discussed in the following sections.

- Preprocessing of the data.
- Selection of significant features used to testing and training the model.
- Partitioning the dataset into five subsets.
- Several classifiers were used to train and test the data and the results were combined using three ensemble approaches, namely bagging, boosting and majority voting. Particle swarm optimization was used to optimize the performance of the model.
- The results of the proposed model were analyzed in terms of accuracy, precision, f1-score and recall.

5.1 Dataset and preprocessing

In the proposed model, NSL-KDD dataset was used which consists of 41 attributes and can be categorized into five classes, namely denial of services (DoS), probe, user to root (U2R) and root to local (R2L) and normal. The dataset consists of 148517 records which are divided into training set (125973 records) and test dataset (22544 records). In the proposed system, the dataset was divided into five subsets and each one of them was trained and tested using different classifier.

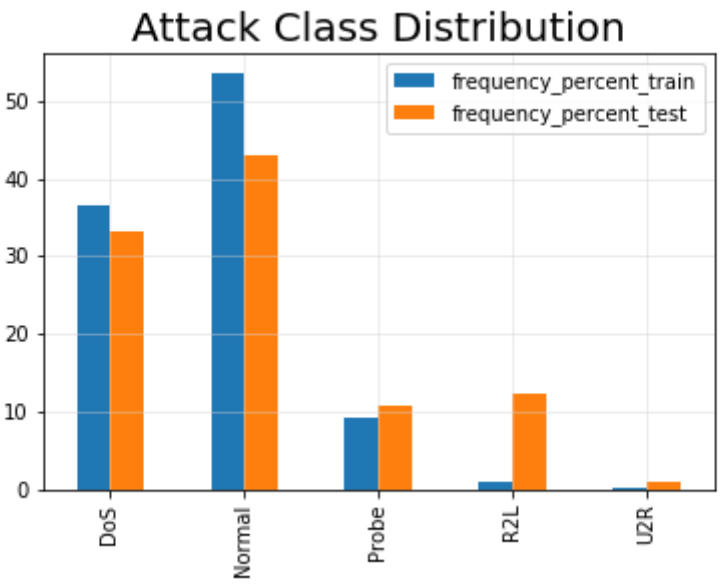

Figure 9: Attack class distribution

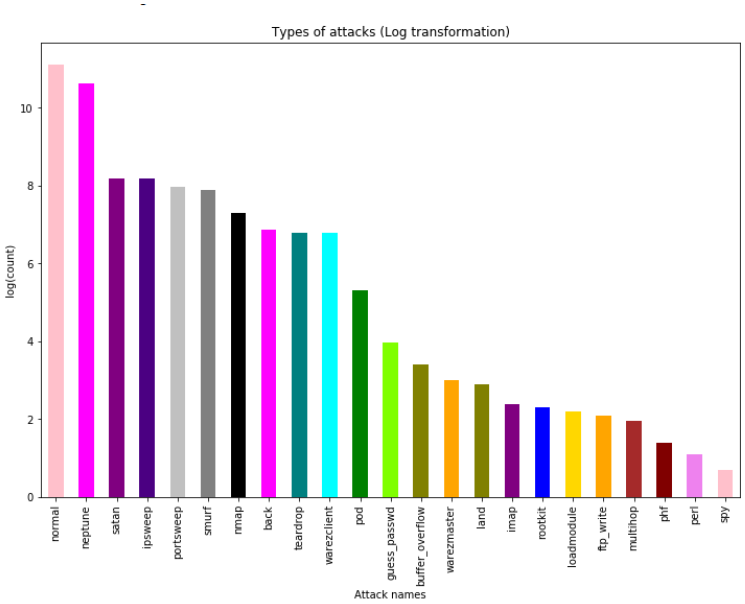

Figure 10: Graphical representation of attacks in NSL-KDD dataset

Pre-processing is a crucial step as it helps in removal of statistical irregularities and selection important features present in the dataset. Those features that are highly correlated to the class distribution are considered to be significant and in the current work, decision tree has been used for feature selection. Prior studies in this field indicate that a proper set of features can improve the performance of the system and reduce the

training time. Initially, certain attribute which have non-numerical value are associated with the records. Since, decision tree that has been employed for feature selection works only on numeric attributes, therefore, these attributes were converted into numeric values in the pre-processing phase. The dataset was then partitioned into subsets and the irrelevant features present in NSL-KDD were removed. Among 41, only ten most significant features were taken account which have been depicted below, depending on which the model was further trained/ tested.

```
selected_features
```

```
['src_bytes',
 'dst_bytes',
 'count',
 'srv_count',
 'diff_srv_rate',
 'dst_host_srv_count',
 'dst_host_same_srv_rate',
 'dst_host_diff_srv_rate',
 'dst_host_rerror_rate',
 'service']
```

Figure 11: Significant features to identify DoS attack

```
selected_features
```

```
['src_bytes',
 'dst_bytes',
 'logged_in',
 'count',
 'srv_count',
 'dst_host_srv_count',
 'dst_host_diff_srv_rate',
 'dst_host_same_src_port_rate',
 'dst_host_serror_rate',
 'service']
```

Figure 12: Significant features to identify Probe attack

```
selected_features
```

```
['duration',
 'src_bytes',
 'dst_bytes',
 'logged_in',
 'count',
 'dst_host_srv_count',
 'dst_host_diff_srv_rate',
 'dst_host_same_src_port_rate',
 'dst_host_serror_rate',
 'service']
```

Figure 13: Significant features to identify R2L attack

```
selected_features
```

```
['src_bytes',
 'dst_bytes',
 'logged_in',
 'count',
 'srv_count',
 'dst_host_srv_count',
 'dst_host_diff_srv_rate',
 'dst_host_same_src_port_rate',
 'dst_host_serror_rate',
 'service']
```

Figure 14: Significant features to identify U2R attack

For feature selection, decision tree was employed which selects the best attribute from a set of attributes based on which the data is partitioned into several classes. It uses a recursive process to classify the records and the process stops when all the records are placed in some class. Since colossal amount of data is transferred in computer networks and decision trees are known to work well for large datasets, they can be effectively used for intrusion detection. Moreover, generalization accuracy and good real time performance of decision trees can be used to detect new attacks.

5.2 Training and testing using ensemble

Individual classifiers which are used to evaluate the performance of the system have their own flaws that lead to compromising the performance of the system. These issues need to

be addressed so that the security of the system is not compromised and more reliable services are provided to the users. For this purpose, ensemble based approach has been employed which is a combination of individual classifier and is more flexible and powerful tool as far as detection of network intrusion is concerned. A good ensemble is the one that maximizes the difference between the base classifier and this can be achieved by dividing the dataset into subsets. The process used in the model as be summarized in the following steps.

- Dividing the dataset into several datasets so that the difference between the various base classifiers is maximized and thus, a better ensemble can be formed.
- Combining the results of the various base classifiers after completion of the training phase so that the final classifier is obtained.
- The output obtained in the training phase is then used to test the data and accordingly the data can be classified as normal or intrusive.

5.3 Results and discussion

In this section, the experimental results for the ensemble based bagging, boosting and majority voting have been discussed. In the current research work, different classifiers viz, support vector machine, K- nearest neighbor, random forest, extra learning tree and multilayer preceptor were used to evaluate the effectiveness of the model on NSL-KDD dataset. To analyze the performance, a comparative study of bagging, boosting and majority voting was conducted. The accuracy of the model is predicted using a confusion matrix which makes use of correctly classified and misclassified entries to analyze the results. Four parameters namely true positive rate, true negative rate, false positive rate and false negative rate are used in confusion matrix to evaluate the efficiency in terms of accuracy, precision, F1 score, support and recall.

- **Accuracy:** Accuracy is computed as the total number of two correct predictions, True Positive (TP) + True Negative (TN) divided by the total number of a dataset Positive (P) + Negative (N).

$$Accuracy = (TP+TN) / (P+N)$$

- **Precision:** Precision is computed as the number of correct positive predictions (TP) divided by the total number of positive predictions (TP + FP). Precision is also known as a positive predictive value.

$$Precision=TP/(TP+FP)$$

- **Recall:** Recall is computed as the number of correct positive predictions (TP) divided by the total number of positives (P). Recall is also known as the true positive rate or sensitivity.

$$Recall=TP/P$$

- **F1 Score:** It is computed as twice the ratio of product of precision and recall to the sum of precision and recall. It is also known as F- measure.

$$F1\ score= 2 * ((precision*recall)/ (precision+ recall))$$

In the proposed system, NSL-KDD dataset was divided into five subsets, the results of the same using bagging ensemble approach have been depicted in form of tables below. The results for defined test environment show superiority of the bagged ensemble technique where five classifiers, namely extra-tree classifier, KNN, random forest, SVM and multi-layer Perceptron were used to evaluate the model performance. The result can be explained by inherent characteristics of bagged approach being more prone to the over-fitting than most of the other evaluated algorithms. It provides high classification accuracy and fast model building procedure. It is capable of learning from highly imbalanced datasets and very small number of attack representatives. It uses bootstrap sampling to reduce the variance and/or improve the predictors accuracy, thus, it was expected to obtain more accurate results.

Table 3: Experimental results of training for bagged Extra-tree classifier

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%

	Support	8082	11547	2087	525	36
Dataset 2	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	8063	11427	2141	607	39
Dataset 3	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	79500	11700	2034	556	37
Dataset 4	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	7886	11704	2065	580	42
Dataset 5	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	8066	11505	2104	568	35

Figure 15: Graphical representation of training for bagged Extra tree classifier

Table 4: Experimental results of testing for bagged Extra-tree classifier

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	99%	99%	97%	97%	100%
	Recall	100%	100%	95%	94%	43%
	F1- score	99%	99%	96%	96%	60%
	Support	13337	19170	3646	913	63
Dataset 2	Accuracy	100%	99%	98%	97%	87%
	Recall	100%	100%	99%	96%	52%
	F1- score	100%	100%	98%	97%	65%
	Support	13337	19170	3646	913	63
Dataset 3	Accuracy	99%	99%	98%	96%	94%
	Recall	99%	100%	95%	94%	52%
	F1- score	99%	99%	96%	95%	67%
	Support	13337	19170	3646	913	63
Dataset 4	Accuracy	99%	99%	97%	97%	90%
	Recall	100%	100%	95%	95%	44%
	F1- score	99%	99%	96%	96%	60%
	Support	13337	19170	3646	913	63
Dataset 5	Accuracy	100%	100%	98%	97%	96%
	Recall	100%	100%	98%	96%	70%
	F1- score	100%	100%	98%	97%	81%
	Support	13337	19170	3646	913	63

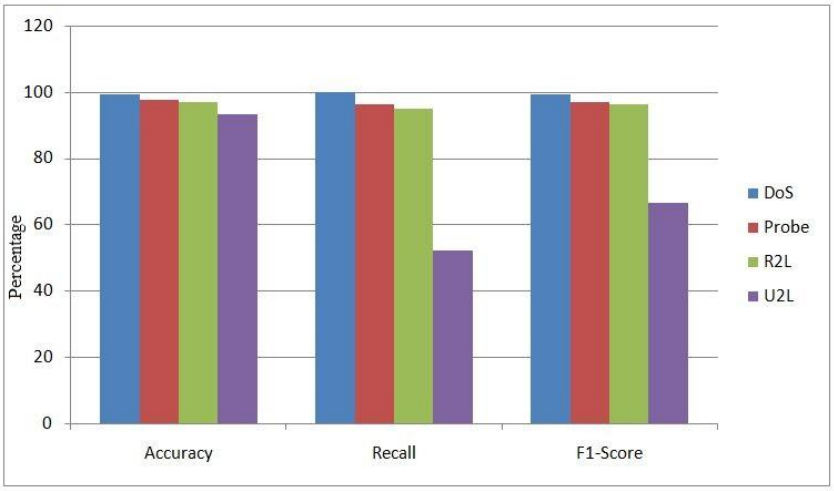

Figure 16: Graphical representation of testing for bagged Extra tree classifier

Table 5: Confusion matrix for Dataset 1 using bagged extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13281	27	29	0	0
Normal	34	19081	50	5	0
Probe	111	60	3458	17	0
R2L	0	44	8	861	0
U2L	0	28	6	2	27

Table 6: Confusion matrix for Dataset 2 using bagged extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13293	42	2	0	0
Normal	12	19118	40	0	0
Probe	5	25	3594	21	1
R2L	0	22	6	881	4
U2L	0	10	14	6	33

Table 7: Confusion matrix for Dataset 3 using bagged extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13270	41	26	0	0
Normal	26	19086	47	11	0
Probe	105	65	3455	20	1
R2L	3	42	5	862	1
U2L	0	19	7	4	33

Table 8: Confusion matrix for Dataset 4 using bagged extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13284	17	36	0	0
Normal	38	19076	49	7	0
Probe	99	57	3473	15	2
R2L	1	39	2	870	1
U2L	0	25	7	3	28

Table 9: Confusion matrix for Dataset 5 using bagged extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13317	12	8	0	0
Normal	0	11505	0	0	0
Probe	0	0	0	0	0
R2L	0	0	0	568	0
U2L	0	0	0	0	35

Table 10: Experimental results of training for bagged K-nearest neighbor

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	98%	99%	96%	93%	79%
	Recall	99%	99%	94%	92%	53%

	F1- score	99%	99%	95%	93%	63%
	Support	8082	11547	2087	525	36
Dataset 2	Accuracy	98%	99%	97%	93%	84%
	Recall	99%	99%	94%	92%	41%
	F1- score	99%	99%	95%	93%	55%
	Support	8063	11427	2141	607	39
Dataset 3	Accuracy	98%	99%	96%	94%	76%
	Recall	99%	99%	93%	93%	35%
	F1- score	99%	99%	95%	94%	48%
	Support	7950	11700	2034	556	37
Dataset 4	Accuracy	98%	99%	97%	94%	76%
	Recall	100%	99%	94%	92%	38%
	F1- score	99%	99%	95%	93%	51%
	Support	7886	11704	2065	580	42
Dataset 5	Accuracy	99%	99%	96%	96%	76%
	Recall	99%	99%	94%	95%	37%
	F1- score	99%	99%	95%	95%	50%
	Support	8066	11505	2104	568	35

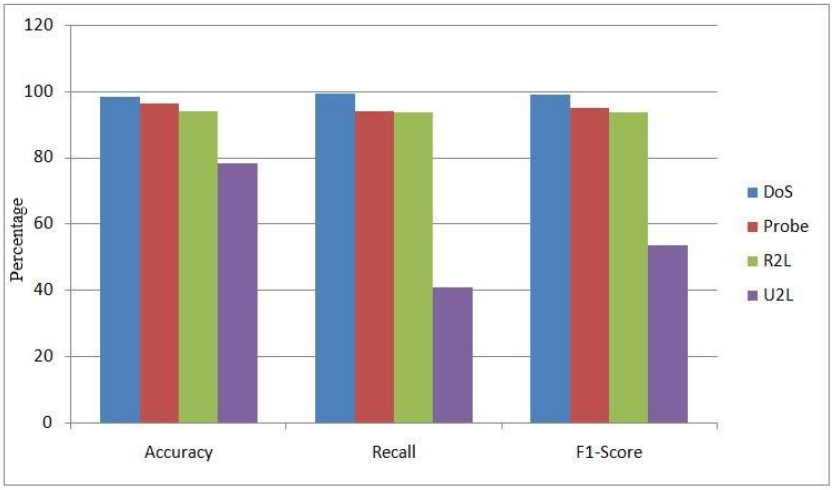

Figure 17: Graphical representation of training for bagged K-nearest neighbor

Table 11: Experimental results of testing for bagged K-nearest neighbor

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	97%	99%	94%	91%	63%
	Recall	99%	99%	91%	89%	41%
	F1- score	98%	99%	92%	90%	50%
	Support	13337	19170	3646	913	63
Dataset 2	Accuracy	98%	98%	95%	89%	81%
	Recall	99%	99%	91%	88%	27%
	F1- score	98%	99%	93%	89%	40%
	Support	13337	19170	3646	913	63
Dataset 3	Accuracy	97%	99%	94%	90%	88%
	Recall	99%	99%	90%	91%	35%
	F1- score	98%	99%	92%	90%	50%
	Support	13337	19170	3646	913	63
Dataset 4	Accuracy	98%	99%	95%	90%	81%
	Recall	99%	99%	91%	90%	33%
	F1- score	98%	99%	93%	90%	47%
	Support	13337	19170	3646	913	63
Dataset 5	Accuracy	98%	99%	95%	93%	90%
	Recall	99%	99%	92%	91%	43%
	F1- score	99%	99%	94%	92%	58%
	Support	13337	19170	3646	913	63

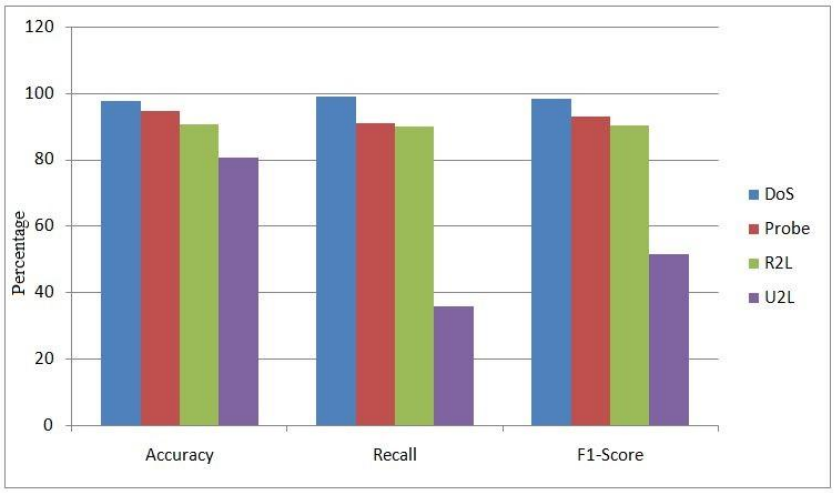

Figure 18: Graphical representation of testing for bagged K-nearest neighbor

Table 12: Confusion matrix for Dataset 1 using bagged K nearest neighbor

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13190	49	93	0	5
Normal	103	18906	105	55	1
Probe	233	68	3315	22	8
R2L	3	88	4	817	1
U2L	1	28	5	3	26

Table 13: Confusion matrix for Dataset 2 using bagged K nearest neighbor

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13207	81	49	0	0
Normal	103	18895	100	71	1
Probe	219	93	3310	21	3
R2L	3	97	7	806	0
U2L	3	25	15	3	17

Table 14: Confusion matrix for Dataset 3 using bagged K nearest neighbor

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13206	44	87	0	0
Normal	100	18910	94	65	1
Probe	243	93	3286	23	1
R2L	8	74	2	828	1
U2L	0	23	14	4	22

Table 15: Confusion matrix for Dataset 4 using bagged K nearest neighbor

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13213	46	78	0	0
Normal	96	18908	98	67	1
Probe	228	84	3312	19	3
R2L	5	78	6	823	0
U2L	0	27	10	5	21

Table 16: Confusion matrix for Dataset 5 using bagged K nearest neighbor

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13240	37	60	0	0
Normal	59	18979	91	40	1
Probe	196	69	3364	15	2
R2L	5	66	7	835	0
U2L	0	26	6	4	27

Table 17: Experimental results of training for bagged Multi-layer Perceptron

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	97%	98%	92%	93%	0
	Recall	97%	99%	89%	84%	0
	F1- score	97%	98%	90%	88%	0

	Support	8082	11547	2087	525	36%
Dataset 2	Accuracy	99%	98%	96%	95%	80%
	Recall	98%	99%	99%	89%	21%
	F1- score	99%	99%	97%	92%	33%
	Support	8063	11427	2141	607	39
Dataset 3	Accuracy	97%	98%	90%	92%	0
	Recall	97%	99%	88%	86%	0
	F1- score	97%	98%	89%	89%	0
	Support	7050	11700	2034	556	37
Dataset 4	Accuracy	100%	98%	96%	96%	89%
	Recall	98%	99%	99%	87%	19%
	F1- score	99%	99%	98%	91%	31%
	Support	7886	11704	2065	580	42
Dataset 5	Accuracy	97%	97%	90%	93%	0
	Recall	97%	99%	88%	83%	0
	F1- score	97%	98%	89%	88%	0
	Support	8066	11505	2104	568	35

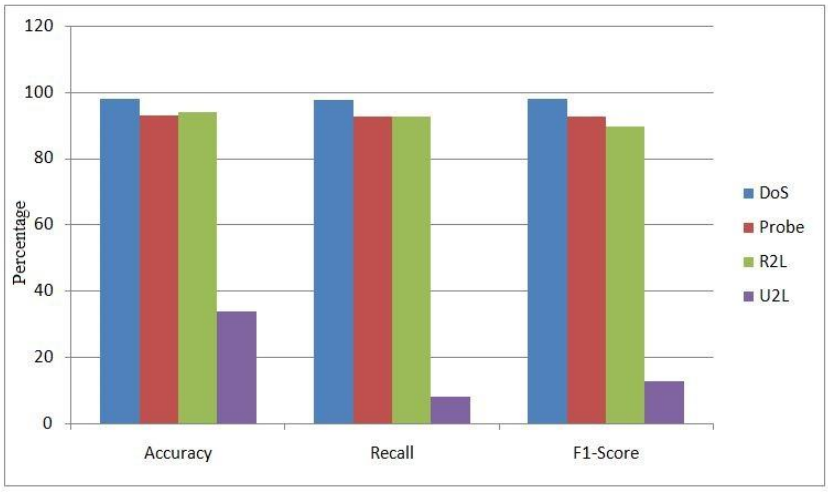

Figure 19: Graphical representation of training for bagged Multi-layer Perceptron

Table 18: Experimental results of testing for bagged Multi-layer Perceptron

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	97%	97%	90%	91%	0
	Recall	97%	99%	88%	82%	0
	F1- score	97%	98%	89%	86%	0
	Support	13337	19170	3646	913	63%
Dataset 2	Accuracy	99%	98%	95%	94%	80%
	Recall	98%	99%	98%	85%	06%
	F1- score	99%	98%	97%	89%	12%
	Support	13337	19170	3646	913	63
Dataset 3	Accuracy	97%	97%	91%	90%	0
	Recall	97%	99%	87%	82%	0
	F1- score	97%	98%	89%	86%	0
	Support	13337	19170	3646	913	63
Dataset 4	Accuracy	99%	98%	96%	95%	100%
	Recall	98%	99%	98%	86%	8%
	F1- score	99%	99%	97%	90%	15%
	Support	13337	19170	3646	913	63
Dataset 5	Accuracy	97%	97%	90%	91%	0
	Recall	97%	99%	88%	81%	0
	F1- score	97%	98%	89%	85%	0
	Support	13337	19170	3646	913	63

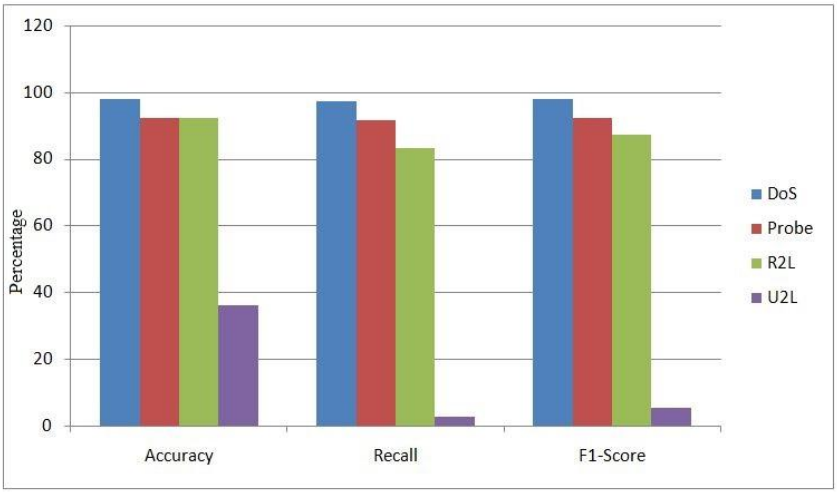

Figure 20: Graphical representation of testing for bagged Multi-layer Perceptron

Table 19: Confusion matrix for Dataset 1 using bagged multi-layer Perceptron

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	12884	275	178	0	0
Normal	94	18905	140	31	0
Probe	345	53	3222	26	0
R2L	0	158	9	746	0
U2L	0	17	30	16	0

Table 20: Confusion matrix for Dataset 2 using bagged multi-layer Perceptron

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13057	235	45	0	0
Normal	82	18985	100	2	1
Probe	0	31	3587	28	0
R2L	0	118	19	776	0
U2L	0	13	28	18	4

Table 21: Confusion matrix for Dataset 3 using bagged multi-layer Perceptron

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	12919	263	155	0	0
Normal	83	18922	130	35	0
Probe	355	73	3188	30	0
R2L	1	150	13	749	0
U2L	0	15	30	18	0

Table 22: Confusion matrix for Dataset 4 using bagged multi-layer Perceptron

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13072	220	45	0	0
Normal	69	19017	76	8	0
Probe	2	34	3591	19	0
R2L	1	116	13	783	0
U2L	0	13	27	18	5

Table 23: Confusion matrix for Dataset 5 using bagged multi-layer Perceptron

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	12879	263	195	0	0
Normal	90	18913	133	34	0
Probe	337	80	3206	23	0
R2L	9	154	13	737	0
U2L	0	15	30	18	0

Table 24: Experimental results of training for bagged Random forest

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	100%	100%	100%	99%	100%
	Recall	100%	100%	99%	99%	72%

	F1- score	100%	100%	99%	99%	84%
	Support	8082	11547	2087	525	36
Dataset 2	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	99%	95%
	F1- score	100%	100%	100%	100%	97%
	Support	8063	11427	2141	607	39
Dataset 3	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	99%	99%	78%
	F1- score	100%	100%	99%	99%	88%
	Support	7950	11700	2034	556	37
Dataset 4	Accuracy	100%	100%	99%	100%	100%
	Recall	100%	100%	99%	98%	79%
	F1- score	100%	100%	99%	99%	88%
	Support	7886	11704	2065	580	42
Dataset 5	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	98%	99%	86%
	F1- score	100%	100%	99%	100%	92%
	Support	8066	11505	2104	568	35

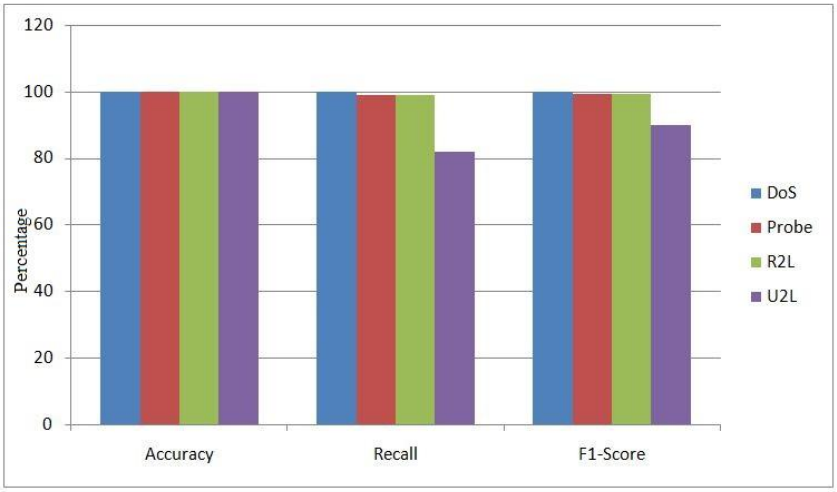

Figure 21: Graphical representation of training for bagged Random forest

Table 25: Experimental results of testing for bagged Random forest

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	99%	99%	97%	96%	96%
	Recall	100%	100%	95%	95%	37%
	F1- score	99%	99%	96%	95%	53%
	Support	13337	19170	3646	913	63
Dataset 2	Accuracy	100%	100%	99%	96%	94%
	Recall	100%	100%	98%	96%	54%
	F1- score	100%	100%	99%	96%	69%
	Support	13337	19170	3646	913	63
Dataset 3	Accuracy	99%	99%	97%	94%	96%
	Recall	100%	100%	94%	93%	43%
	F1- score	99%	99%	96%	94%	59%
	Support	13337	19170	3646	913	63
Dataset 4	Accuracy	99%	99%	97%	96%	96%
	Recall	100%	100%	94%	94%	35%
	F1- score	99%	99%	96%	95%	51%
	Support	13337	19170	3646	913	63
Dataset 5	Accuracy	99%	99%	97%	96%	100%
	Recall	100%	100%	95%	94%	29%
	F1- score	99%	99%	96%	95%	44%
	Support	13337	19170	3646	913	63

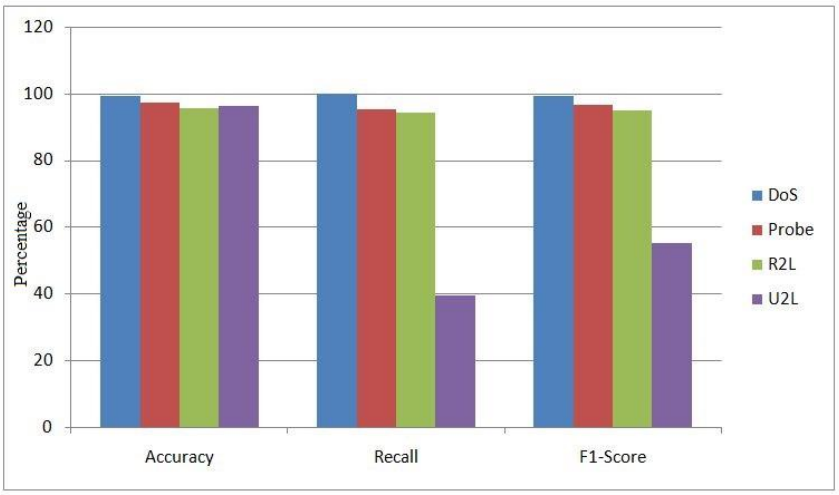

Figure 22: Graphical representation of testing for bagged Random forest

Table 26: Confusion matrix for Dataset 1 using bagged random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13272	36	28	0	1
Normal	34	19075	53	8	0
Probe	122	55	3448	21	0
R2L	0	46	4	863	0
U2L	0	29	6	5	23

Table 27: Confusion matrix for Dataset 2 using bagged random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13300	32	5	0	0
Normal	6	19127	37	0	0
Probe	21	19	3584	22	0
R2L	1	25	5	880	2
U2L	0	11	7	11	34

Table 28: Confusion matrix for Dataset 3 using bagged random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13273	42	22	0	0
Normal	23	19076	49	22	0
Probe	115	61	3442	28	0
R2L	4	47	10	851	1
U2L	0	23	9	4	27

Table 29: Confusion matrix for Dataset 4 using bagged random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13271	30	36	0	0
Normal	30	19075	51	14	0
Probe	123	57	3444	22	0
R2L	2	43	7	860	1
U2L	1	26	10	4	22

Table 30: Confusion matrix for Dataset 5 using bagged random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13277	26	34	0	0
Normal	28	19081	50	11	0
Probe	112	53	3463	18	0
R2L	0	37	14	862	0
U2L	2	28	11	4	18

Table 31: Experimental results of training for bagged support vector machine

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	94%	93%	80%	93%	0
	Recall	92%	98%	77%	33%	0

	F1- score	93%	95%	79%	49%	0
	Support	8082	11547	2087	525	36
Dataset 2	Accuracy	95%	91%	75%	92%	0
	Recall	90%	98%	78%	15%	0
	F1- score	93%	94%	76%	26%	0
	Support	8063	11427	2141	607	39
Dataset 3	Accuracy	94%	92%	79%	90%	0
	Recall	92%	98%	77%	27%	0
	F1- score	93%	95%	78%	41%	0
	Support	7950	11700	2034	556	37
Dataset 4	Accuracy	94%	93%	81%	90%	0
	Recall	92%	98%	79%	34%	0
	F1- score	93%	95%	80%	50%	0
	Support	7886	11704	2065	580	42
Dataset 5	Accuracy	90%	91%	78%	84%	0
	Recall	91%	97%	57%	56%	0
	F1- score	91%	93%	66%	67%	0
	Support	8066	11505	2104	568	35

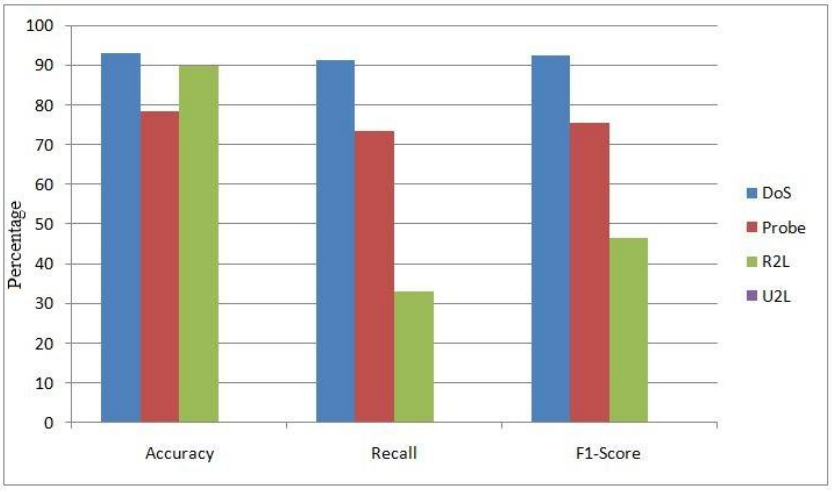

Figure 23: Graphical representation of training for bagged support vector machine

Table 32: Experimental results of testing for bagged support vector machine

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	94%	93%	80%	92%	0
	Recall	92%	98%	78%	30%	0
	F1- score	93%	95%	79%	46%	0
	Support	13337	19170	3646	913	63
Dataset 2	Accuracy	95%	91%	74%	88%	0
	Recall	91%	97%	76%	12%	0
	F1- score	93%	94%	75%	22%	0
	Support	13337	19170	3646	913	63
Dataset 3	Accuracy	94%	92%	81%	85%	0
	Recall	92%	98%	77%	24%	0
	F1- score	93%	95%	79%	37%	0
	Support	13337	19170	3646	913	63
Dataset 4	Accuracy	94%	93%	82%	89%	0
	Recall	93%	98%	79%	32%	0
	F1- score	93%	95%	80%	47%	0
	Support	13337	19170	3646	913	63
Dataset 5	Accuracy	90%	91%	78%	82%	0
	Recall	91%	96%	57%	51%	0
	F1- score	91%	94%	66%	63%	0
	Support	13337	19170	3646	913	63

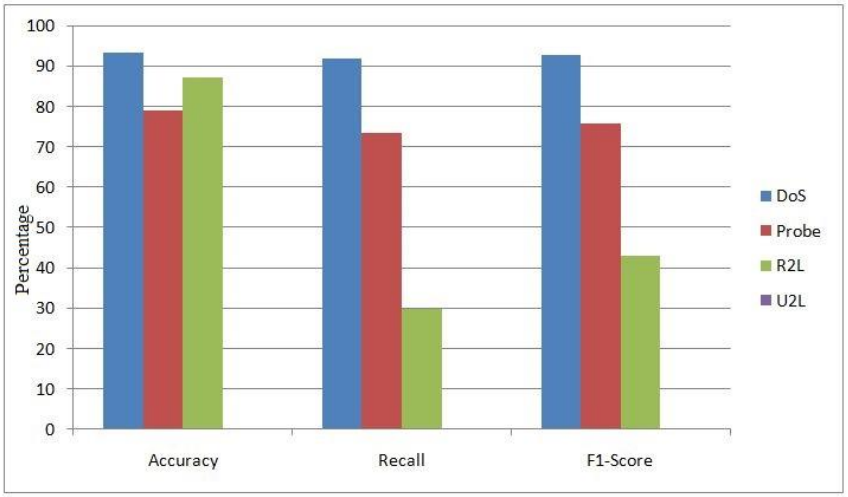

Figure 24: Graphical representation of testing for bagged support vector machine

Table 33: Confusion matrix for Dataset 1 using bagged support vector machine

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	12320	634	383	0	0
Normal	187	18707	263	13	0
Probe	617	186	2834	9	0
R2L	7	577	52	277	0
U2L	5	32	24	2	0

Table 34: Confusion matrix for Dataset 2 using bagged support vector machine

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	12111	688	538	0	0
Normal	135	18675	357	3	0
Probe	548	302	2786	10	0
R2L	0	754	45	114	0
U2L	0	32	29	2	0

Table 35: Confusion matrix for Dataset 3 using bagged support vector machine

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	12331	644	362	0	0
Normal	180	18725	246	19	0
Probe	609	216	2803	18	0
R2L	6	657	33	217	0
U2L	4	32	25	2	0

Table 36: Confusion matrix for Dataset 4 using bagged support vector machine

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	12345	659	333	0	0
Normal	189	18734	230	17	0
Probe	610	150	2868	18	0
R2L	7	573	42	291	0
U2L	4	32	25	2	0

Table 37: Confusion matrix for Dataset 5 using bagged support vector machine

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	12183	838	316	0	0
Normal	455	18475	183	57	0
Probe	907	612	2084	43	0
R2L	5	384	59	465	0
U2L	0	30	29	4	0

The results of the boosting ensemble approach using extra-tree classifier and random forest have been depicted in form of tables below. Accordingly, the confusion metric representing the results of the test dataset in form of accuracy, recall, F1-score and support have been presented.

Table 38: Experimental results of training for boosted extra-tree classifier

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	8082	11547	2087	525	36
Dataset 2	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	8063	11427	2141	607	39
Dataset 3	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	795%0	11700	2034	556	37
Dataset 4	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	7886	11704	2065	580	42
Dataset 5	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	8066	11505	2104	568	35

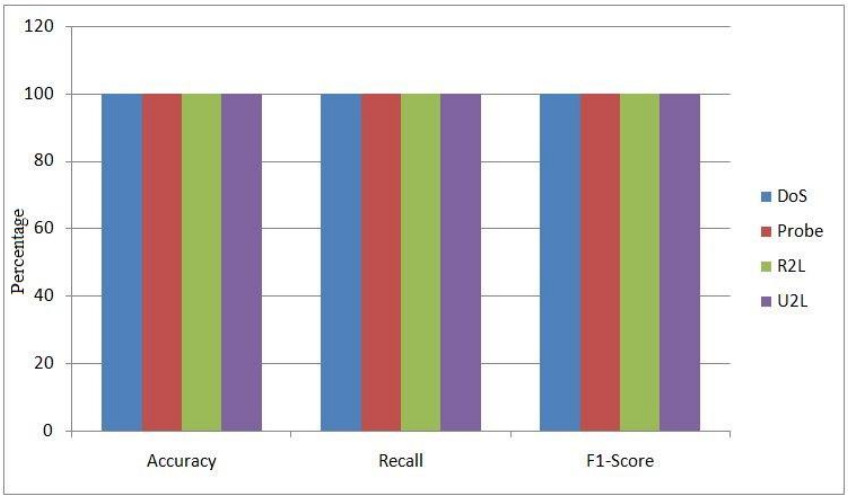

Figure 25: Graphical representation of training for boosted Extra tree classifier

Table 39: Experimental results of testing for boosted extra-tree classifier

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	99%	99%	97%	97%	93%
	Recall	100%	99%	94%	94%	44%
	F1- score	99%	99%	96%	95%	60%
	Support	13337	19170	3646	913	63
Dataset 2	Accuracy	99%	99%	98%	96%	91%
	Recall	100%	100%	94%	94%	46%
	F1- score	99%	99%	96%	95%	61%
	Support	13337	19170	3646	913	63
Dataset 3	Accuracy	99%	99%	97%	96%	93%
	Recall	99%	100%	94%	93%	60%
	F1- score	99%	99%	96%	94%	73%
	Support	13337	19170	3646	913	63
Dataset 4	Accuracy	99%	99%	97%	97%	88%
	Recall	99%	99%	94%	95%	48%
	F1- score	99%	99%	96%	96%	62%
	Support	13337	19170	3646	913	63
Dataset 5	Accuracy	100%	100%	98%	98%	91%
	Recall	100%	100%	98%	95%	76%

	F1- score	100%	100%	98%	97%	83%
	Support	13337	19170	3646	913	63

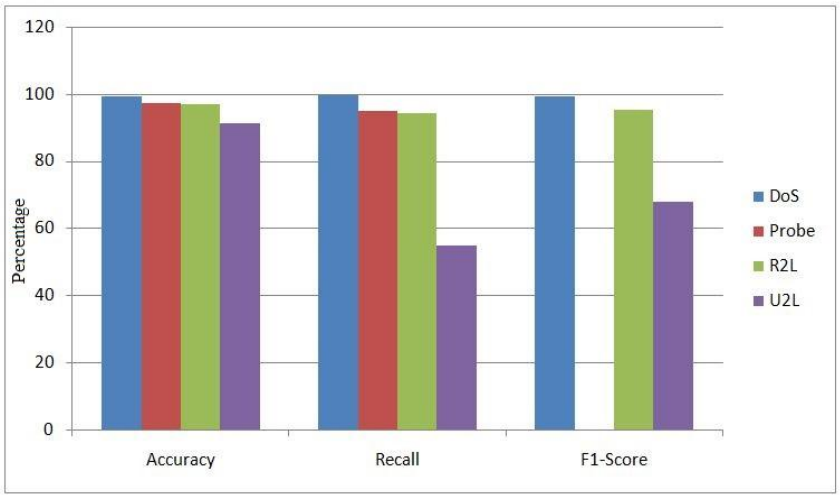

Figure 26: Graphical representation of testing for boosted Extra tree classifier

Table 40: Confusion matrix for Dataset 1 using boosted extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13279	19	37	2	0
Normal	51	19067	42	10	0
Probe	135	66	3434	10	1
R2L	2	45	9	856	1
U2L	0	26	6	3	28

Table 41: Confusion matrix for Dataset 2 using boosted extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13273	30	34	0	0
Normal	29	19089	39	13	0
Probe	118	71	3439	17	1
R2L	1	46	3	861	2
U2L	1	18	12	3	29

Table 42: Confusion matrix for Dataset 3 using boosted extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13268	35	34	0	0
Normal	29	19079	46	16	0
Probe	125	72	3427	20	2
R2L	5	46	8	853	1
U2L	0	15	6	4	38

Table 43: Confusion matrix for Dataset 4 using boosted extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13269	22	46	0	0
Normal	49	19061	46	12	2
Probe	131	56	3442	16	1
R2L	3	39	2	868	1
U2L	4	21	6	2	30

Table 44: Confusion matrix for Dataset 5 using boosted extra-tree classifier

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13321	6	9	1	0
Normal	20	19114	32	4	0
Probe	25	46	3565	10	0
R2L	1	24	14	869	5
U2L	0	8	3	4	48

Table 45: Experimental results of training for boosted random forest

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%

	F1- score	100%	100%	100%	100%	100%
	Support	8082	11547	2087	525	36
Dataset 2	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	8063	11427	2141	607	39
Dataset 3	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	7950	11700	2034	556	37
Dataset 4	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	7886	11704	2065	580	42
Dataset 5	Accuracy	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1- score	100%	100%	100%	100%	100%
	Support	8066	11505	2104	568	35

Figure 27: Graphical representation of training for boosted random forest

Table 46: Experimental results of testing for boosted random forest

Expert	Parameters	Dos	Normal	Probe	R2L	U2R
Dataset 1	Accuracy	100%	100%	98%	97%	100%
	Recall	100%	100%	98%	96%	44%
	F1- score	100%	100%	98%	96%	62%
	Support	13337	19170	3646	913	63
Dataset 2	Accuracy	100%	100%	99%	97%	93%
	Recall	100%	100%	99%	97%	67%
	F1- score	100%	100%	99%	97%	78%
	Support	13337	19170	3646	913	63
Dataset 3	Accuracy	99%	99%	98%	95%	100%
	Recall	100%	100%	96%	95%	56%
	F1- score	99%	99%	96%	95%	71%
	Support	13337	19170	3646	913	63
Dataset 4	Accuracy	99%	99%	98%	97%	92%
	Recall	100%	100%	96%	95%	38%
	F1- score	99%	99%	97%	96%	54%
	Support	13337	19170	3646	913	63
Dataset 5	Accuracy	100%	100%	98%	97%	97%
	Recall	100%	100%	98%	97%	54%
	F1- score	100%	100%	98%	97%	69%
	Support	13337	19170	3646	913	63

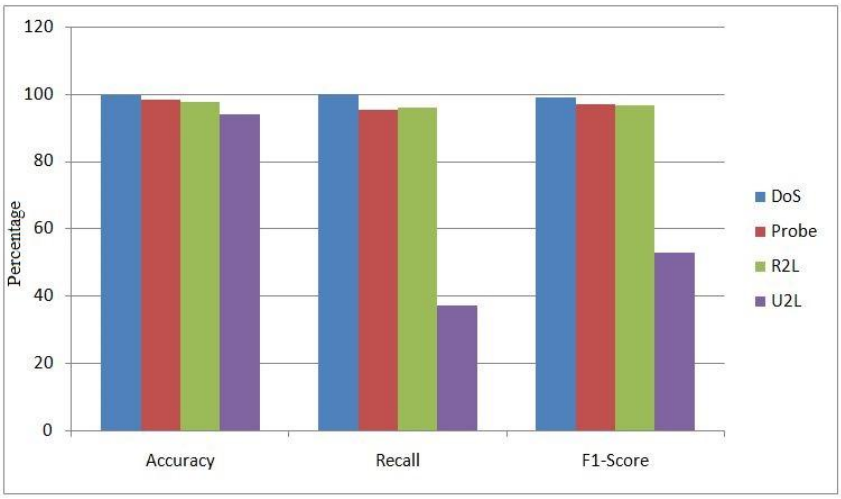

Figure 28: Graphical representation of testing for boosted random forest

Table 47: Confusion matrix for Dataset 1 using boosted random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13313	22	2	0	0
Normal	19	19105	45	1	0
Probe	23	21	3585	17	0
R2L	0	22	12	879	0
U2L	0	17	5	13	28

Table 48: Confusion matrix for Dataset 2 using boosted random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13312	25	0	0	0
Normal	10	19127	31	2	0
Probe	4	21	3606	15	0
R2L	0	17	3	890	3
U2L	0	5	8	8	42

Table 49: Confusion matrix for Dataset 3 using boosted random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13275	37	25	0	0
Normal	18	19083	51	19	0
Probe	93	49	3482	22	0
R2L	3	33	8	869	0
U2L	0	16	6	6	35

Table 50: Confusion matrix for Dataset 4 using boosted random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13291	21	25	0	0
Normal	30	19093	42	5	0
Probe	83	41	3503	18	1
R2L	1	39	2	870	1
U2L	0	28	8	3	24

Table 51: Confusion matrix for Dataset 5 using boosted random forest

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13312	25	0	0	0
Normal	10	19127	31	2	0
Probe	4	21	3606	15	0
13312	0	17	3	890	3
U2L	0	5	8	8	42

The results of the majority voting have been depicted in form of tables where extra-tree classifier, support vector machine, KNN, multi-layer Perceptron and random forest were used in combination to enhance the intrusion detection rate. Accordingly, the confusion metric representing the results of the test dataset in form of accuracy, recall, F1-score and support have been presented.

Table 52: Experimental results of training using majority voting

Expert	Parameters	Normal	DoS	Probe	R2L	U2R
Dataset 1	Accuracy	99%	99%	98%	97%	85%
	Recall	100%	99%	95%	95%	47%
	F1- score	99%	99%	97%	96%	61%
	Support	8082	11547	2087	525	36
Dataset 2	Accuracy	100%	99%	98%	98%	100%
	Recall	100%	99%	96%	97%	44%
	F1- score	99%	100%	97%	97%	61%
	Support	11427	8063	2141	607	39
Dataset 3	Accuracy	99%	99%	98%	98%	92%
	Recall	100%	100%	95%	96%	30%
	F1- score	99%	100%	97%	97%	45%
	Support	11700	7950	2034	556	37
Dataset 4	Accuracy	99%	99%	98%	97%	93%
	Recall	100%	100%	96%	96%	31%
	F1- score	99%	99%	97%	96%	46%
	Support	11704	7886	2065	580	42
Dataset 5	Accuracy	99%	99%	97%	98%	100%
	Recall	99%	100%	95%	96%	34%
	F1- score	99%	99%	96%	97%	51%
	Support	11505	8066	2104	568	35

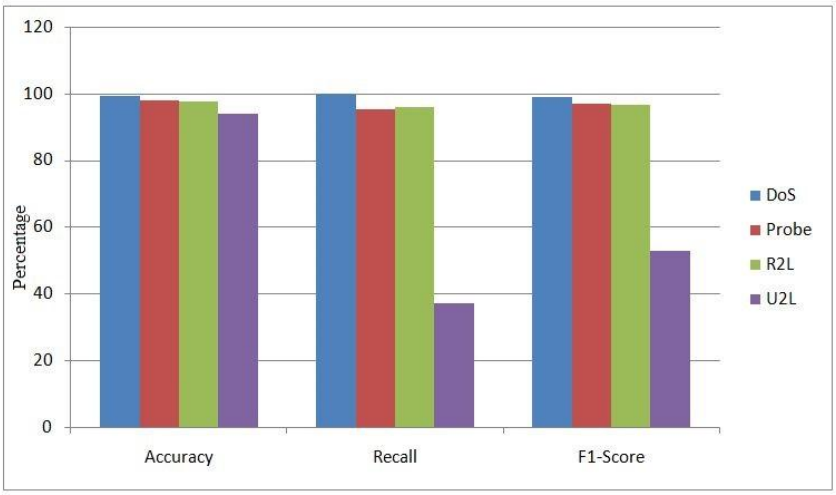

Figure 29: Graphical representation of training for majority voting

Table 53: Experimental results of testing using majority voting

Expert	Parameters	Normal	DoS	Probe	R2L	U2R
Dataset 1	Accuracy	99%	98%	96%	94%	73%
	Recall	99%	99%	93%	92%	38%
	F1- score	99%	99%	94%	93%	50%
	Support	13337	19170	3646	913	63
Dataset 2	Accuracy	99%	98%	96%	95%	100%
	Recall	99%	99%	92%	94%	27%
	F1- score	99%	99%	94%	94%	42%
	Support	19170	13337	3646	913	63
Dataset 3	Accuracy	99%	98%	96%	95%	100%
	Recall	99%	99%	92%	93%	29%
	F1- score	99%	99%	94%	94%	44%
	Support	19170	13337	3646	913	63
Dataset 4	Accuracy	99%	98%	96%	95%	86%
	Recall	99%	99%	92%	93%	29%
	F1- score	99%	99%	94%	94%	43%
	Support	19170	13337	3646	913	63
Dataset 5	Accuracy	99%	98%	97%	96%	95%

	Recall	99%	99%	93%	94%	30%
	F1- score	99%	99%	95%	95%	46%
	Support	19170	13337	3646	913	63

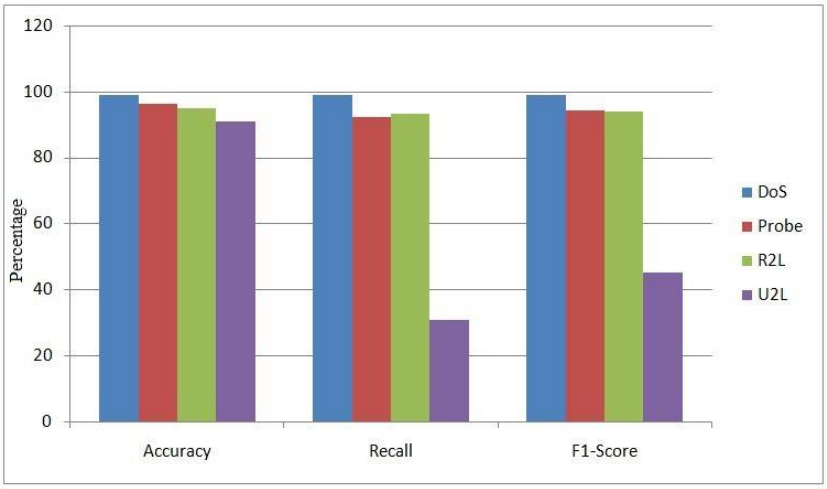

Figure 30: Graphical representation of testing for majority voting

Table 54: Confusion matrix for Dataset 1 using majority voting

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13285	27	23	1	1
Normal	49	19064	52	5	0
Probe	132	63	3442	9	0
R2L	0	48	10	854	1
U2L	0	29	6	3	25

Table 55: Confusion matrix for Dataset 2 using majority voting

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13262	45	30	0	0
Normal	29	19088	45	7	1
Probe	126	64	3442	13	1
R2L	2	43	3	865	0

U2L	1	18	12	2	30

Table 56: Confusion matrix for Dataset 3 using majority voting

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13275	36	26	0	0
Normal	32	19084	42	12	0
Probe	124	62	3443	16	1
R2L	5	44	3	861	0
U2L	1	19	8	4	31

Table 57: Confusion matrix for Dataset 4 using majority voting

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13274	27	36	0	0
Normal	45	19066	50	9	0
Probe	112	66	3449	15	4
R2L	4	49	3	856	1
U2L	1	22	9	3	28

Table 58: Confusion matrix for Dataset 5 using majority voting

Confusion Matrix	Dos	Normal	Probe	R2L	U2R
DoS	13272	23	40	1	1
Normal	45	19076	49	8	0
Probe	127	62	3446	11	0
R2L	5	40	15	853	0
U2L	1	16	9	3	34

Through the experimental results, we have demonstrated that classification accuracy can be improved by combining the opinions from the multiple experts into one using an ensemble approach. In case of bagging, extra-tree classifier outperformed other classifiers

with an average model accuracy of 99.1232% whereas the worst performance was obtained for bagged support vector machine with an accuracy of 91.2192%. Likewise, for bagged KNN, MLP and random forest, an average accuracy of 97.7472%, 97.0378% and 98.924% was achieved respectively. For boosting based ensemble approach, model accuracy of 99.3096% and 98.9018% was attained for random forest and extra-tree classifiers respectively. Similarly, in case of majority voting, an average accuracy of 98.7928% was obtained on NSL-KDD dataset. Hence, bagged extra-tree classifier outperforms the other ensemble techniques. Thus, it is reliable and can be effectively used for network intrusion detection.

Chapter 6

Conclusion and Future scope

6 Conclusion and Future Scope

Intrusion detection and prevention are essential to current and future networks and information systems, because our daily activities are heavily dependent on them. Furthermore, future challenges will become more daunting because of the Internet of Things. In this respect, intrusion detection systems have been important in the last few decades. Several techniques have been used in intrusion detection systems, but machine learning techniques are common in the recent literature. Additionally, different machine learning techniques have been used, but some techniques are more suitable for analyzing huge data for intrusion detection of network and information systems. To address this problem, the latest ensemble based machine learning techniques have been analyzed to present a brief overview about the work being conducted in the field of intrusion detection. The aim of the current research work is to present an effective intrusion detection system and for the same ensemble based technique was employed. A well dataset, NSL-KDD was used for performance evaluation which was partitioned into five subsets and each subset was trained/ tested separately. For improving the training speed and reducing the computational complexity, data was preprocessed so that only relevant features could be attained. Several classifiers were used for training/ testing and majority voting was used to obtain the final results. Particle swarm optimization was used for improving the performance of the model. The final results showed a good performance and thus, the proposed model could effectively used for network intrusion detection.

There is a scope of improvement in terms of feature selection, i.e., an ensemble of feature selection techniques can be used to select significant features so that feature selection technique can be optimized. For enhancing the reliability of the model, various other classifiers can be invested in future. Moreover, as machine learning algorithm have certain vulnerabilities, attackers can use to same to launch the attacks and thus, there is a need to focused in this direction and design a classifier that can withstand the relevant security issues. A hybridization approach with various different models can also be considered in order to overcome their individual limitations and improve the performance of the model from their complementary features.

References

7 References

[1] Aburomman AA, Ibne Reaz M Bin. A novel SVM-kNN-PSO ensemble method for intrusion detection system. Appl Soft Comput J 2016;38:360–72.

[2] Hosseini Bamakan SM, Wang H, Yingjie T, Shi Y. An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization. Neurocomputing 2016;199:90–102.

[3] Jabbar MA, Aluvalu R, Reddy SS. RFAODE: A novel ensemble intrusion detection system. Procedia Comput Sci 2017;115:226–34.

[4] Timčenko V, Gajin S. Ensemble classifiers for supervised anomaly based network intrusion detection. Proc. - 2017 IEEE 13th Int. Conf. Intell. Comput. Commun. Process. ICCP, 2017, p. 13–9.

[5] Ma T, Wang F, Cheng J, Yu Y, Chen X. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. Sensors 2016;16:1–23.

[6] Kumar Singh Gautam R, Doegar EA. An ensemble approach for intrusion detection system using machine learning algorithms. 8th Int. Conf. Cloud Comput. Data Sci. Eng. Conflu. 2018, 2018, p. 61–4.

[7] Sahu SK, Katiyar A, Kumari KM, Kumar G, Mohapatra DP. An SVM-based ensemble approach for intrusion detection. Int J Inf Technol Web Eng 2019;14:66–84.

[8] Chen W, Kong F, Mei F, Yuan G, Li B. A novel unsupervised anomaly detection approach for intrusion detection system. IEEE 3rd Int. Conf. Big Data Secur. cloud, 2017, p. 69–73.

[9] Rajasekaran M, Ayyasamy A. A novel ensemble approach for effective intrusion detection system. 2017 Second Int. Conf. Recent trends challenges Comput. Model. ICRTCCM, 2017, p. 244–50.

[10] Adeniran AA, Akinola SO. An ensemble data mining approach for intrusion detection in a computer network. Int J Sci Eng Investig 2017;6:73–7.

[11] Salo F, Nassif AB, Essex A. Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. Comput Networks 2019;148:164–75.

[12] Pham NT, Foo E, Suriadi S, Jeffrey H, Lahza HFM. Improving performance of

intrusion detection system using ensemble methods and feature selection. ACM Int. Conf. proceeding Ser., 2018, p. 1–6.

[13] Zhou Y, Cheng G, Jiang S, Dai M. An efficient intrusion detection system based on feature selection and ensemble classifier. J Latex Cl Files 2015;14:1–12.

[14] Wang W, Li Y, Wang X, Liu J, Zhang X. Detecting android malicious apps and categorizing benign apps with ensemble of classifiers. Futur Gener Comput Syst 2018;78:987–94.

[15] Tama BA, Rhee KH. A combination of PSO-based feature selection and tree-based classifiers ensemble for intrusion detection systems. Adv. Comput. Sci. ubiquitous Comput., 2015, p. 489–95.

[16] Kanakarajan NK, Muniasamy K. Improving the accuracy of intrusion detection using GAR-forest with feature selection. Proc. 4th Int. Conf. Front. Intell. Comput. Theory Appl. 2015. Adv. Intell. Syst. Comput., 2016, p. 539–47.

[17] Haq NF, Onik AR, Shah FM. An ensemble framework of anomaly detection using hybridized feature selection approach (HFSA). SAI Intell. Syst. Conf., 2015, p. 989–95.

[18] Osanaiye O, Cai H, Choo KKR, Dehghantanha A, Xu Z, Dlodlo M. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. Eurasip J Wirel Commun Netw 2016;130:1–10.

[19] Amiri F, Rezaei Yousefi M, Lucas C, Shakery A, Yazdani N. Mutual information-based feature selection for intrusion detection systems. J Netw Comput Appl 2011;34:1184–99.

[20] Gao W, Hu L, Zhang P, He J. Feature selection considering the composition of feature relevancy. Pattern Recognit Lett 2018;112:70–4.

[21] Ahmad I, Basheri M, Iqbal MJ, Rahim A. Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection. IEEE Access 2018;6:33789–95. doi:10.1109/ACCESS.2018.2841987.

[22] Fawagreh K, Gaber MM, Elyan E. Random forests: From early developments to recent advancements. Syst Sci Control Eng 2014;2:602–9.

[23] Li W, Yi P, Wu Y, Pan L, Li J. A new intrusion detection system based on KNN classification algorithm in wireless sensor network. J Electr Comput Eng 2014:1–

8.

[24] Kennedy J, Eberhart R. Particle swarm optimization. Encycl. Mach. Learn., 2011, p. 1942–8. doi:10.1109/TST.2016.7442504.

[25] Wang X, Yang J, Teng X, Xia W, Jensen R. Feature selection based on rough sets and particle swarm optimization. Pattern Recognit Lett 2007;28:459–71.

[26] Menahem E, Shabtai A, Rokach L, Elovici Y. Improving malware detection by applying multi-inducer ensemble. Comput Stat Data Anal 2009;53:1483–94.

[27] Arun Raj Kumar P, Selvakumar S. Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems. Comput Commun 2013;36:303–19.

[28] Folino G, Sabatino P. Ensemble based collaborative and distributed intrusion detection systems: A survey. J Netw Comput Appl 2016;66:1–16.

[29] Breiman L. Bagging predictions. Mach Learn 1996;24:123–40.

[30] Lazarevic A, Kumar V. Feature bagging for outlier detection. Proceeding Elev. ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '05, 2005, p. 157–66.

[31] Gaikwad DP, Thool RC. Intrusion detection system using bagging ensemble method of machine learning. Proc. - 1st Int. Conf. Comput. Commun. Control Autom. ICCUBEA, 2015, p. 291–5.

[32] Engen V. Machine Learning for network based intrusion detection: An investigation into discrepancies in findings with the KDD Cup '99 data set and multi-objective evolution of neural network classifier ensembles for imbalanced data. 2010.

[33] Syarif I, Zaluska E, Prugel-Bennett A, Wills G. Application of bagging, boosting and stacking to intrusion detection. 8th Int. Conf. Mach. Learn. data Min., 2012, p. 593–602.

[34] Govindarajan M. Evaluation of ensemble classifiers for intrusion detection 2016;10:1045–53.

[35] RE M, Valentini G. Ensemble methods : A review. Data Min. Mach. Learn. Astron. Appl., 2012, p. 1–40.

[36] Schapire RE. The strength of weak learnability. Mach Learn 1990;5:197–227. doi:10.1023/A:1022648800760.

[37] Farid DM, Rahman MZ, Rahman CM. Adaptive intrusion detection based on boosting and Nave Bayesian classifier adaptive intrusion detection based on boosting and Naïve Bayesian classifier. Artic Int J Comput Appl 2011;24:12–9.

[38] Gudadhe M, Prasad P, Kapil Wankhade L. A new data mining based network intrusion detection model. 2010 Int. Conf. Comput. Commun. Technol., 2010, p. 731–5.

[39] Maclin R, Opitz D. An empirical evaluation of bagging and boosting for artificial neural networks. Fourteenth Natl. Conf. Artif. Intell., 1997, p. 546–551.

[40] Kuncheva LI, Alpaydin E. Combining Pattern Classifiers: Methods and Algorithms. vol. 18. 2007. doi:10.1109/tnn.2007.897478.

[41] Panda M, Patra MR. Ensemble voting system for anomaly based network intrusion detection. Int J Recent Trends Eng 2009;2:8–13.

[42] Lueckenga J, Engel D, Green R. Weighted vote algorithm combination technique for anomaly based Smart Grid intrusion detection systems. Int. Jt. Conf. Neural Networks, 2016, p. 2738–42.