CHAPTER 1

INTRODUCTION

1.1 Overview

The importance of securing information in an organization has become of utmost importance. No organization would want its information to be known by other organizations especially not its competitors. In order to protect their information, organizations are willing to spend millions of dollars, showing how important the issue is. One of the solutions to securing information is by using the Intrusion Detection System (IDS). Simply put, IDS that can be in the form of an application or device, just like firewall, would detect malicious or suspicious activities in the network. IDS was first introduced in 1980 by Anderson and then improved by Denning in 1987.

Basically, there are two Intrusion Detection techniques, i.e. Anomaly Detection and Misuse Detection. Anomaly Detection is basically based on assumptions that attacker behavior is different from normal user's behavior. The strategy is to look for unusual or abnormal activities in a system or network. When the detection is performed, the normal behavior data will be compared to actual user's data. If the offset is below threshold value, then user's behavior can be considered as normal without any intention of attack. One of the advantages of this detection is that it has high detection rate and able to detect novel attack.

On the other hand, Misuse Detection (also known as signature-based detection), uses pattern matching. In order to determine an attack, it will compare the data with the signature in signature database, and if the data match with the pattern as in signature database, then it will define as attack. This type of detection has high detection rate with low false alarm. In this project, the first technique i.e. the Anomaly Based Detection is used. The Anomaly Based Detection can be implemented using different other techniques such as Statistical Model, Knowledge Based Model and Machine Learning. In this project we are using Machine Learning.

1.2 Brief Description

As the Internet continues to enjoy widespread use in our day to day lives, a variety of attacks, such as Denial of Service (DoS), Probing, Spam and so on, emerge on a regular basis. IDS is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a management station. Intrusion detection system (IDS) is an effective method to guard against malicious attacks and has been widely used. Intrusion detection and prevention systems are primarily focused on identifying possible incidents, logging information about them, and reporting attempts. It attracts many research interests and we have attempted to add to this large research effort. Our project is an attempt to reduce false alarm rate for Intrusion Detection System by using Machine Learning algorithm. We aim to design IDS by using machine learning which can meet the demands of Reducing False Alarm Rate with higher detection rate. Many types of IDS already exist in the world which provides assistance at different stages of project development. But a problem commonly observed is the high False Alarm Rate. Our software should be able to assist the developers in this department greatly along with the individual stage support.

1.3 Problem Definition

The title of our project is “Pattern Based Security Using Machine Learning Techniques.”

Our project is an attempt to reduce false alarm rate for Intrusion Detection System by using Machine Learning algorithm. We aim to design IDS by using machine learning which can meet the demands of Reducing False Alarm Rate with higher detection rate. Many types of IDS already exist in the world that provides assistance at different stages of project development. But a problem commonly observed is the high False Alarm Rate. Our software should be able to assist the developers in this department greatly along with the individual stage support.

1.4 Applying software engineering approach

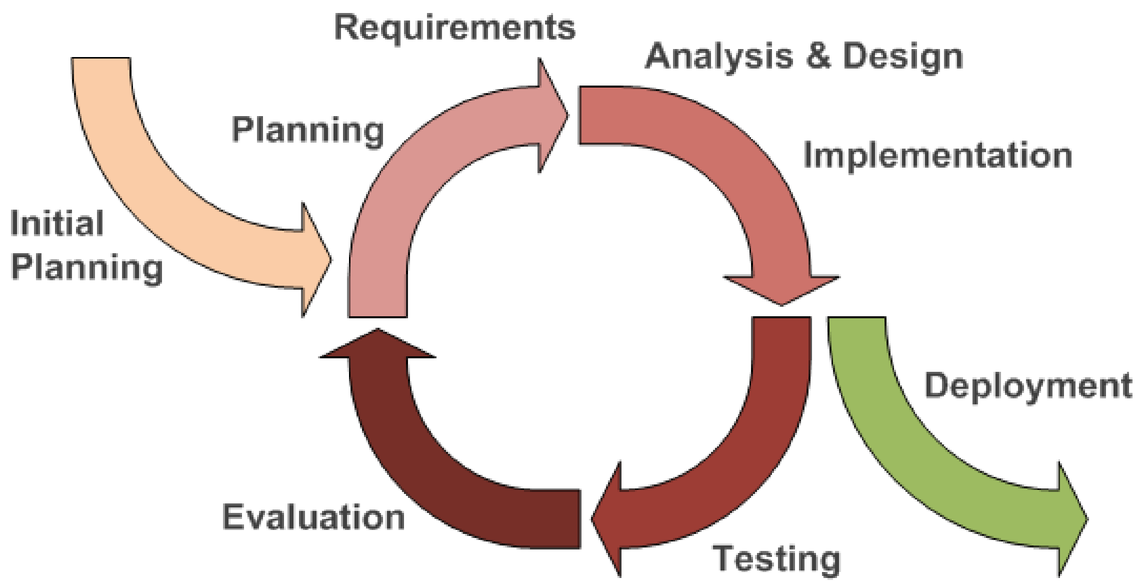


Figure 1.1 Incremental Approach

In this procedure first the planning on the project is done. After that all the requirements are gathered for the project. In analysis and design first started with small modules to create. In case of failure the procedure is repeated again and again. After the implementation of small modules big modules are implemented. It is tested with many test cases and after all evaluation it is deployed.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

The section Literature Survey helps us to understand the fundamental concepts of the project. The need of the project is explained under Motivation and Goals. Various architecture and tools needed are explained later. Then, explained is the detailed study of the Applications and the Existing System.

2.2 Research Papers

1. "Improving the Accuracy of Intrusion Detection Systems by Using the combination of Machine Learning Approaches", Hadi Sarvari, and Mohammad Mehdi Keikha. [9]

This paper is focused on combination of machine learning approaches as to detect the system attacks. This article proposes a combinatory system. Primarily, it was concluded that the system accuracy increases by increasing the number of classes. Then it was revealed that this increase continues to a point at which it stops and remains constant.
They are given description about KDDCup99 Data set. One of the features of KDD 99 is that their samples are much fewer than those of other classes and since machine learning-based systems do not learn the features of these two classes, their detection accuracy are also much less than other two classes.

They are presented the Intrusion Detection Systems by Combination of Machine Learning system in that they are given results first of separately for NN, Decision Tree and SVM algorithm. After that they gave results on the different combinations of algorithms. The U2R and R2L class samples are much fewer than other classes and therefore their detection percentage is less than of the others, too. This is called unbalanced data problem with input data of different classes. They generated data for these two classes to solve the problem of the unbalanced data, so that they helped the system to recognize the features of these two classes. The addition of such a data may produce data not being in the input data. In fact, unknown features have been introduced to the system. As a whole, this system improved the recognition of R2L and U2R classes. They are given results on the basis of experiment performed by them. KDD 99 contains 24 different known attacks in training data and 14 unknown attacks in testing data. After analyzing different combinatory models, they reached their conclusion that the best model is one containing SVM, DT, 1NN, 2NN and 3NN.

2. “Comparison of Machine Learning Algorithm Performance in Detecting Network Intrusion”, Kamarularifin Abd Jalil, and Mohamad Noorman Masrek. [10]

In this paper they are given some basic information about intrusion detection system and need of IDS in current time. In this paper, three Machine Learning algorithms namely Neural Network (NN), Support Vector Machine (SVM) and Decision Tree (DT) have been compared in terms accuracy, detection rate, false alarm rate and accuracy for four categories of attack under different percentage of normal data. Neural Network is a mathematical model or computational model that simulates the structure functional aspects of biological Neural Network. The techniques of Neural Networks follow the same theories of how human brain works. Support Vector machines are the most common and popular method for machine learning tasks in classification and regression. By using this algorithm, a model can predict whether a new example falls into one categories or other. Decision tree algorithm is used for classification problem. In this algorithm, the data set is learnt and modeled. Therefore, whenever a new data item is evaluated, it will be classified accordingly. One of the strength of Decision Tree is it can works well with huge data sets as well as in real-time Intrusion Detection. The purpose of this research is to determine the best algorithm that can be used as benchmarks for research in Intrusion Detection by using KDD 99 dataset. KDD 99 dataset was the current benchmark dataset in Intrusion Detection. However, the dataset was distributed unevenly and might produce an error if only one set of dataset is used. Therefore in this research, the dataset was distributed evenly and the datasets from training and testing were combined. The main reason they combine the dataset is to make sure that all 39 attacks in both datasets can be run simultaneously with different percentage of normal data in order to get an average value.

They are given the behavior of attacks and normal data as True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The findings of this study suggested that these three machine-learning algorithms would misclassify an undesirably large amount of data when the number of attack is higher than the number normal data. From the experiment conducted, they found that, in accuracy and detection rate, the Decision Tree (J48) was superior from others. However in determine the false alarm rate; Support Vector Machines has outperformed the others. On the other hand, the comparison result of accuracy for four categories of attack shows that, the Decision Tree algorithm performed much better than other two algorithms.

3. "Evaluating machine learning algorithms for detecting network intrusions", Mrutyunjaya Panda, and Manas Ranjan Patra. [11]

In this paper they mainly focused on detecting network intrusions. They employ ensemble algorithms in modeling network intrusion detection systems, to improve detection performance. In this they described the methods employed in their proposed framework and given how to apply these methods to build an efficient intrusion detection system model. They are given overview of the framework and ensemble learning methods like AdaBoost, Random Forest, Naïve Bayes. They presented Naïve Bayes algorithm also in order to compare their earlier results on the proposed method, to find its suitability in building an efficient network intrusion detection model.

They are given experimental results on the basis of data sets, performance measurements and error rate. Results on the network audit data shows that AdaBoost is not suitable for building network intrusion detection model, while there is a compromise between the Naïve Bayes and Random forest. Several intrusion detection schemes for detecting network intrusions are proposed in this paper. When applied to KDDCup'99 data set, developed algorithms for learning classifiers were successful in detecting network attacks than standard data mining techniques based on neural networks.

4. "A Detailed Analysis of the KDD CUP 99 Data Set", Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. [12]

In this paper, they found after studying all KDD CUP 99 dataset that the first important deficiency in the KDD data set is the huge number of redundant records. Analyzing KDD train and test sets, they found that about 78% and 75% of the records are duplicated in the train and test set, respectively. This large amount of redundant records in the train set will cause learning algorithms to be biased towards the more frequent records, and thus prevent it from learning un frequent records which are usually more harmful to networks such as U2R attacks.

The analysis showed that there are two important issues in the data set which highly affects the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches. To solve these issues, they have proposed a new data set, NSL-KDD, which consists of selected records of the complete KDD data set.
They are given the description of KDD CUP 99 dataset. This data set is prepared by Stolfo et al. and is built based on the data captured in DARPA'98 IDS evaluation program. DARPA'98 is about 4 gigabytes of compressed raw (binary) tcpdump data of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with about 100 bytes. The two weeks of test data have around 2 million connection records. KDD training dataset consists of approximately 4,900,000 single connection vectors each of which contains 41 features and is labeled as either normal or an attack, with exactly one specific attack type. The attacks are classified into four types DOS, U2R, R2L and Probe. KDD'99 features can be classified into three groups: Basic features, Traffic features and Content features.

5. "Practical real-time intrusion detection using machine learning approaches", Phurivit Sangkatsanee, Naruemon Wattanapongsakorn, Chalermpol Charnsripinyo [13]

The growing prevalence of network attacks is a well-known problem which can impact the availability, confidentiality, and integrity of critical information for both individuals and enterprises. In this paper, we propose a real-time intrusion detection approach using a supervised machine learning technique. Our approach is simple and efficient, and can be used with many machine-learning techniques. We applied different well-known machine learning techniques to evaluate the performance of our IDS approach. Our experimental results show that the Decision Tree technique can outperform the other techniques. Therefore, we further developed a real-time intrusion detection system (RT-IDS) using the Decision Tree technique to classify on-line network data as normal or attack data. We also identified 12 essential features of network data that are relevant to detecting network attacks using the information gain as our feature selection criterions. Our RT-IDS can distinguish normal network activities from main attack types (Probe and Denial of Service (DoS)) with a detection rate higher than 98% within 2 s. We also developed a new post-processing procedure to reduce the false-alarm rate as well as increase the reliability and detection accuracy of the intrusion detection system.

6. "A survey on intrusion detection techniques", Sandip Sonawane, Shailendra Pardeshi and Ganesh Prasad. [14]

Intrusion detection system (IDS) is a security layer that is used to discover ongoing intrusive attacks and anomaly activities in information systems and is usually working in a dynamically changing environment. Although increasing IDSs are developed in the literature, network security administrators are faced with the task of analyzing enormous alerts produced from the analysis of different event streams. In this paper we present three types of intrusion detection based on the source of detection – host based, network based and hybrid intrusion detection and also focuses on intrusion detection techniques i.e. misuse detection and anomaly detection techniques, supervised and unsupervised based learning based on the different approaches.

7. "A Comprehensive Analysis and study in Intrusion Detection System using Data Mining Techniques", G.V. Nadiammai, S.Krishnaveni, M. Hemalatha. [15]

Data mining refers to extracting knowledge from large amounts of data. Most of the current systems are weak at detecting attacks without generating false alarms. Intrusion detection systems (IDSs) are increasingly a key part of system defense. An intrusion can be defined as any set of actions that compromise the integrity, confidentiality or availability of a network resource (such as user accounts, file system, kernels & so on).Data mining plays a prominent role in data analysis. In this paper, classification techniques are used to predict the severity of attacks over the network. I have compared zero R classifier, Decision table classifier & Random Forest classifier with KDDCUP 99 databases from MIT Lincoln Laboratory.

8. "Results of the KDD'99 Classifier Learning", Charles Elkan. [16]

In this paper the author has given the information about the contest taken in KDD'99 conference. The task for the classifier-learning contest organized in conjunction with the KDD'99 conference was to learn a predictive model (i.e. a classifier) capable of distinguishing between legitimate and illegitimate connections in a computer network. In total 24 entries were submitted for the contest. It is important to note that the data quality issue affected only the labels of the test examples. The training data was unaffected, as was the unlabeled test data. In this he gave the winning entries. The difference in performance between the three best entries is only of marginal statistical significance. There is given a confusion matrix for performance of the winning entries. The top-left entry in the confusion matrix shows that 60262 of the actual "normal" test examples were predicted to be normal by this entry. The last column indicates that in total 99.5% of the actual "normal" examples were recognized correctly. The bottom row shows that 74.6% of test examples said to be "normal" were indeed "normal" in reality. It is difficult to evaluate exactly the statistical significance of differences between entries. However it is important not to read too much into differences that may well be statistically insignificant, i.e. due to randomness in the choice of training and test examples. If the threshold for statistical significance is taken to be two standard errors, then the winning entry is significantly superior to all others except the second and third best. He is given that the simple method performs well. Most participants achieved results no better than those achievable with very simple methods. According to its author, one entry was simply "the trusty old 1-nearest neighbor classifier."

9. “Pattern Based Network security using Decision Trees and Support Vector Machine”, Pachghare V.K., Kulkarni P. [17]

Decision tree is an important method for both induction research and data mining, which is mainly used for model classification and prediction. WEKA is software which is capable of doing work on various decision tree algorithms and support vector machine. In this paper, the comparative study of all Decision Tree algorithms is done. The training time, Accuracy and size of tree are the parameters used as performance measures. It is concluded that J48 Graft algorithm performs better than other algorithms. The dataset used is KDD cup'99 dataset. This dataset contains normal as well as abnormal packets. The dataset is highly uneven. We worked out on some 1000 selected packets. The support vector machine algorithms have the ability to be trained and `learn' in a given environment. This feature can be used in connection with an intrusion detection system, where the support vector machine algorithm can be trained to detect intrusions by recognizing patterns of an intrusion. This paper outlines an investigation on the support vector machine models and choice of one of them for evaluation and implementation. The work also includes works on computer networks, providing a description and analysis of the structure of the computer network in order to generate network features.

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Introduction

Organizations have come to realize that network security technology has become very important in protecting its information. With tremendous growth of internet, attack cases are increasing each day along with modern attack method. One of the solutions to this problem is by using Intrusion Detection System (IDS). An Intrusion detection system is designed to classify the system activities into normal and abnormal. We will use combination of machine learning approaches to detect the system attacks from the normal ones.

Basically there are two types of Intrusion detection techniques, i.e. Anomaly Detection and Misuse Detection. Anomaly Detection is basically based on assumption that attacker behavior is different from normal user's behavior. The strategy is to look for unusual or abnormal activities in a system or network. When the detection is performed, the normal behavior data will be compared to actual user's data. If the offset is below threshold value, then user's behavior can be considered as normal without any intention of attack. One of the advantage of this detection is that it has high detection rate and able to detect novel attack.

On the other hand, Misuse Detection, also known as signature based detection uses pattern matching. In order to determine an attack, it will compare the data with the signature in the signature database, and if the data match with the pattern as in signature database, then it will define as attack.

The behavior of attacks and normal data can be described as below:

1) True Positive (TP): when amount of attack data detected is actual attack data.

2) True Negative (TN): when amount of normal data detected is actually normal data.

3) False Positive (FP): when normal data is detected as attack data.

4) False Negative (FN): when attack data is detected as normal data.

The main objective of an Intrusion Detection System is to have high accuracy and detection rate with low false alarm.

3.1.1 Purpose

The main purpose of our project is to provide platform and boost our work implemented to the ones who wish to study and do the research in the network security and intrusion detection system as well as contribute to the field of new innovations in intrusion detection system. This document can also be a reference to developers who wants to increase the detection rate of intrusion that is ultimate aim of this project.

3.1.2 Intended audience and reading suggestions

This document is meant for researchers, developers, testers, and documentation writers. The SRS document aims to explain in an easy manner, the basic idea behind the 'To reduce False Alarm Rate (FAR) in Intrusion Detection System (IDS) through machine learning algorithm' and how the developers aim to achieve their goals. It also aims to introduce to the users the main features of packets and major attacks such as Dos, Probe, U2r, R2l and what is the Classification of attacks.

3.1.3 Project Scope

Basic functionality of IDS can be differentiated with regard to the detection technique, misuse and anomaly detection (behavior).

1. Misuse detection- It is also known as signature detection technique which searches for well-known patterns.

2. Anomaly detection-It is also known as behavior detection technique that builds a model of the normal network behavior and attacks can be detected by measuring significant deviation of the current status against the behavior expected from the model.

Therefore, anomaly-based systems are able to detect new and yet unknown threats. IDS is a network based anomaly detector aimed to provide accurate and real-time enterprise intrusion detection and prevention solution to combat known attacks and reduce False Alarm Rate (FAR). IDS is developed to provide a complete, better than existing and an open source solution to the rising number of insecure enterprise networks. Scope of our project is to analyze our IDS and provide our case study as a reference for further research and development on IDS.

This projects aims to:

1. Achieve maximum intrusion detection rate.

2. Achieve negligible false alarm rates.

3. Increase efficiency of the system.

4. Provide availability of resources for further research and development.

The project outlines the following objectives:

1. Achieving maximum detection and false alarm rates.

2. Providing a user-friendly menu for configuring and scaling the available options.

3. Smooth running of system with complete error handling.

3.1.4 Design and Implementation Constraints

a) **Processing Power**: IDS require high speed data capturing, analysis, detection and prevention within negligible time. With these features, high speed processing machine is required to fulfill all the tasks.

b) **Deployment Point**: We will directly feed data from KDD cup99 to IDS.

c) **Detection/False Alarm Rates**: Detection and false alarm rates depend on the choice of algorithm from the user. With detections, come a number of false alarms as well. There will be further release of IDS in future that might improve these parameters.

d) **Operating Platform**: IDS will work for several distributions of Linux and Windows.

3.1.5 Assumptions and Dependencies

We assume that the data to be feed in the training and testing phase of the system is provided from KDD Cup99 that is in specific and standard format so that it can be processed correctly.

3.2 System Features

The proposed solution shall provide several services to its users. Major services provided by the IDS system are briefly discussed below.

3.2.1 Detecting Attacks

KDDCup99 dataset is widely used in the experiment of IDS as it provides the basis for comparison of different approaches that require large datasets. We are using KDDCup99 as our dataset. This dataset was generated based on 1998 DARPA/MIT Lincoln Lab original datasets. It represents the activities at US Air Force local area network (LAN), which have normal traffic and malicious activities that were injected in the datasets. It has 4GB of compressed binary dump data of 7 weeks network traffic. The traffic represents 5 million connection records with 100 bytes of size each. There are four simulated attacks that were injected in the datasets.

1. Denial of Service (dos): Attacker tries to prevent legitimate users from using a service, e.g. neptune, teardrop, etc.
2. Remote to Local (r2l): Attacker does not have an account on the victim machine, hence tries to gain access, e.g. guess-passwd, spy, etc.

3. User to Root (u2r): Attacker has local access to the victim machine and tries to gain super user privileges, e.g. buffer-overflow, perl, etc.

4. Probe: Attacker tries to gain information about the target host, e.g. portsweep, satan, etc.

Table 3.1 Feature categories in kdd99 dataset

Categories	Features
TCP basic features (1~9)	duration, protocol type, service, flag, src_bytes, dst_bytes, land, wrong fragment, urgent
TCP content features (10~22)	hot,num_failed_logins, logged_in,num_compromised, root_shell, su_attempted,num_root, num_file_creations, num_shells,num_access_files, num_outbound_cmds,is_hot_login, is_guest_login
TCP Traffic features (23~31)	count, srv_count, serror_rate, srv_seror_rate,rerror_rate, rv_rerror_rate, same_srv_rate,diff_srv_rate, srv_diff_host_rate
Host-Based Network Traffic (32~41)	dst_host_count, dst_host_srv_count,dst_host_same_srv_rate, dst_host_diff_srv_rate,dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate,dst_host_srv_serror_rate, dst_host_srv_serror_rate,dst_host_rerror_rate, dst_host_srv_rerror_rate

The purpose of intrusion detection is to detect malicious activities from the given datasets. So, how to exactly express an illegal connection is essential. In this paper, the selection of input features depend on attacks type. There are the majority of DOS attacks that frequently scan the hosts (or ports) using a shorter time interval than 2 seconds. As a result, these attacks produce intrusion patterns with time-based traffic features. With regard to Probe attacks, some are like to DOS attacks, and the others scan the hosts (or ports) using a larger time interval than 2 seconds, for example, one in every minute. So, host-based and time-based traffic features are used as Probe attacks patterns. However the R2L and U2R attacks, unlike most of the DOS and Probe attacks, don't have any "intrusion only" frequent sequential patterns. This is because the DOS and Probe attacks involve many connections to some host(s) in a very short period of time, the R2L and Probe attacks are embedded in the data portions of the packets, and normally involves only a single connection. Therefore, content features were used as these attacks features patterns. No matter what attacks type, basic features are included in intrusion feature patterns, so the results of selection to input features are shown in Table 3.2.

Table 3.2 The selection result of input features

Attack Type	Selection of Features	Size
Dos	Basic features+ time-based traffic features	18
Probe	Basic features+ (time-based+ host-based) traffic features	28
R2L	Basic features+ content features	22
U2R	Basic features+ content features	22

The experiment includes two parts. The first part uses 41-features multiple classified intrusion detection system, and tests its performance. The other carries out the features from above table and contrasts the results of the two groups experiment. In the experiments, the training data set and testing data set come from the data set for KDD CUP 99 competition. The training data set includes four sub-sets, namely, DOS training sample sub-set, Probe training sample sub- set, R2L training sample sub-set and U2R training sample sub-set. Every sub-set compose the only one type attack samples with normal samples, for example, the R2L training sample sub-set includes only R2L attack sample and normal sample. Every training sample is labeled as either attack (the value is 1) or normal (the value is 0). The sample data is shown in table 3.2.

3.2.2 Displaying Result

We will display our results in different format such as:

1. Tabular Formats

2. Graphical Format

3. Percentage Format

In tabular format the result will be displayed with actual and correctly identified number of packets of all types of attacks. Number of packets taken for training and number of packets taken for testing will be shown. In graphical format results will be displayed in a Chart whereas in percentile results will be simply calculated in percentage and displayed.

3.3 External Interface Requirements

3.3.1 User Interface

A user interface is available for providing following functionalities:

1. Display result in Table view.

2. Display result in Graphical view.

3. Display result in Line Chart view.

There will be different graphical view like line chart and bar graph. The program in JAVA will act as our front end in the system. The starting page will allow user to input data in .arff format.

3.3.2 Hardware Interface

The solution makes extensive use of several hardware devices. These devices include

- 2 GB RAM or more.
- Core-2-duo or more advance processors. For faster and efficient work the above requirements should be fulfilled.

3.3.3 Software Interfaces

- Java 1.6
- Windows 7

3.4 Nonfunctional Requirements

3.4.1 Performance Requirements

The solution has to exhibit following performance requirements.

1. The system must have very high detection rates.

2. The system must have very low false alarm rates.

These requirements shall be achieved by using a combination of several algorithms. Another performance requirement is the detection of anomalies in real time. The active anomaly detection module is proposed for the same purpose.

3.4.2 Safety Requirements

There are no specific safety requirements associated with the proposed system. The IDS is composed of well-known and commonly used Machine learning technique that does not cause any safety hazards.

3.4.3 Security Requirements

Only authorized personnel are allowed to use the product and go through selection procedures. In case of forgotten passwords contact the developers. Similarly, changing the features of the solutions at runtime also requires password based authentication.

3.4.4 Software Quality Attributes

• **Reliability:** IDS should provide reliability to the user that the product will run stably with all the features mentioned above available and executing perfectly. It should be tested and debugged completely. All exceptions should be well handled.

• **Accuracy:** IDS should be able to reach the desired detection level. It should generate minimum false positive alerts with maximum detection rate.

• Resources: IDS should use minimal resources in terms of memory, time and CPU.

• User Friendliness: IDS should have a graphical user interface with user-friendly menu.

3.5 Analysis Models

3.5.1 Data Flow Diagrams

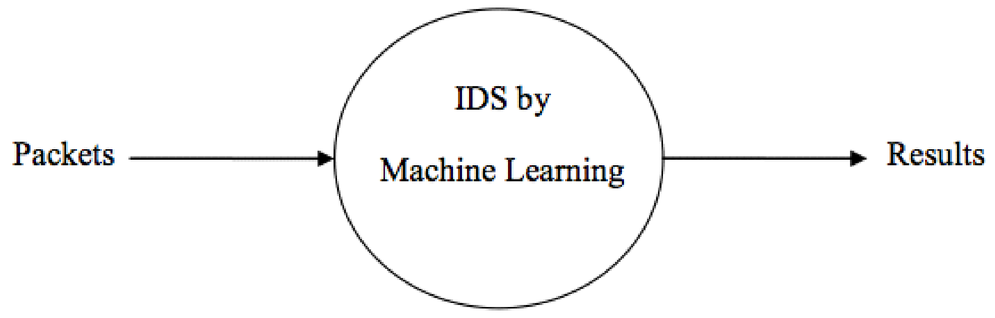


Fig 3.1 DFD Level-0

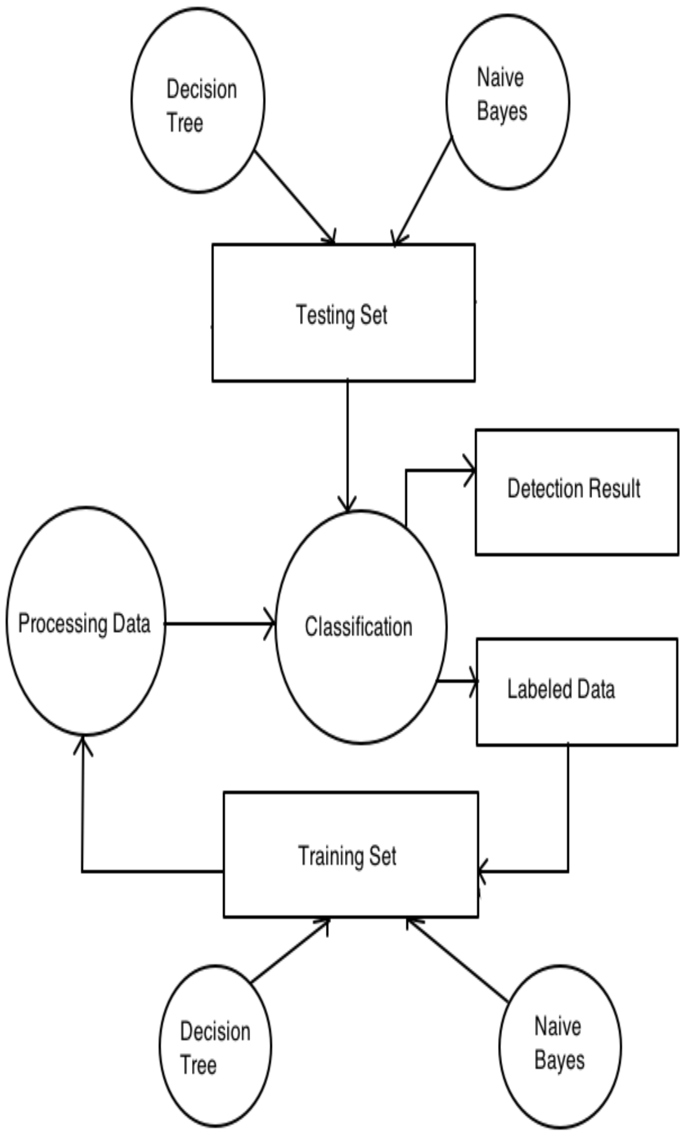


Fig.3.2 DFD Level-1

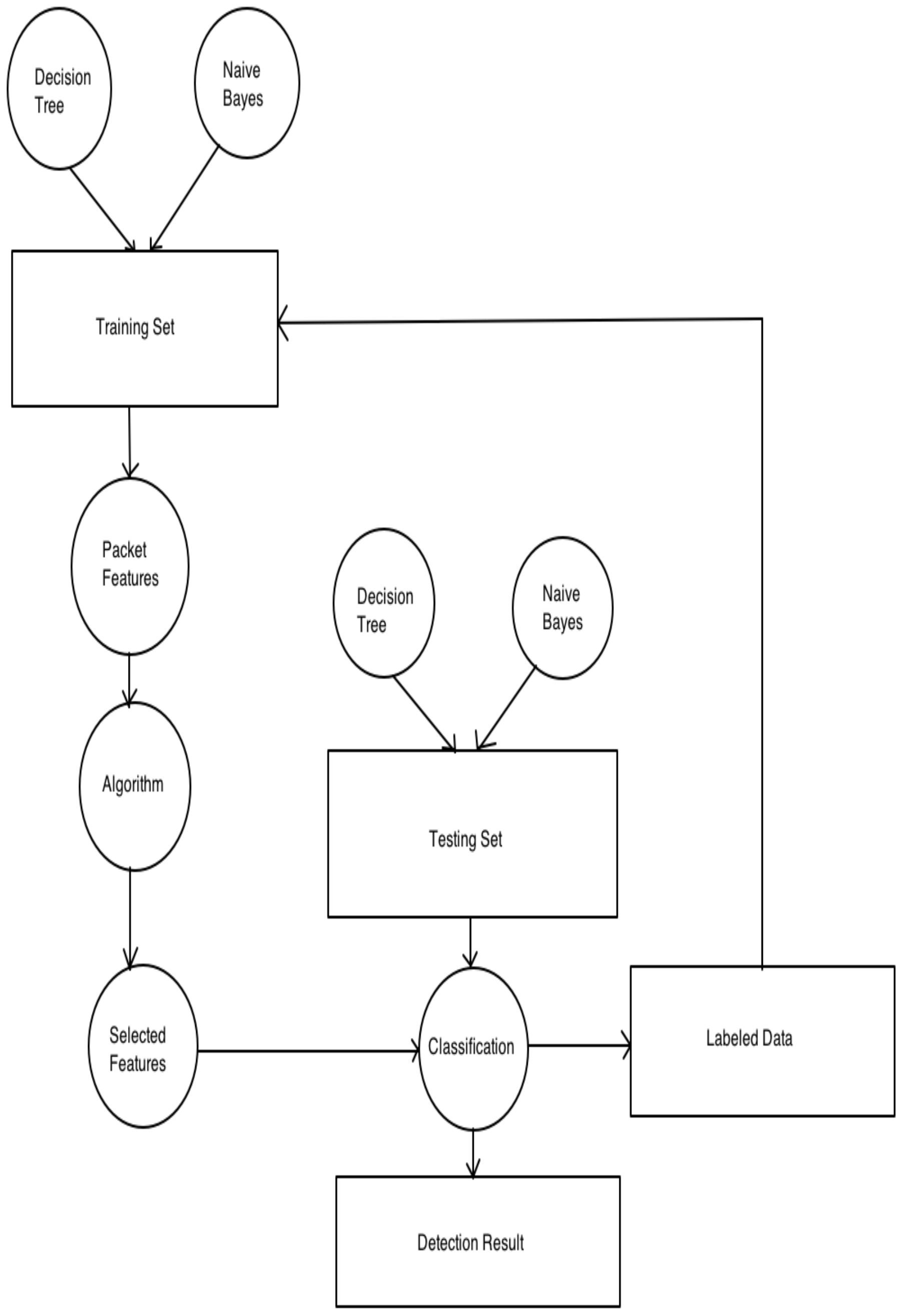


Fig.3.3 DFD Level-2

3.5.2 Class Diagram

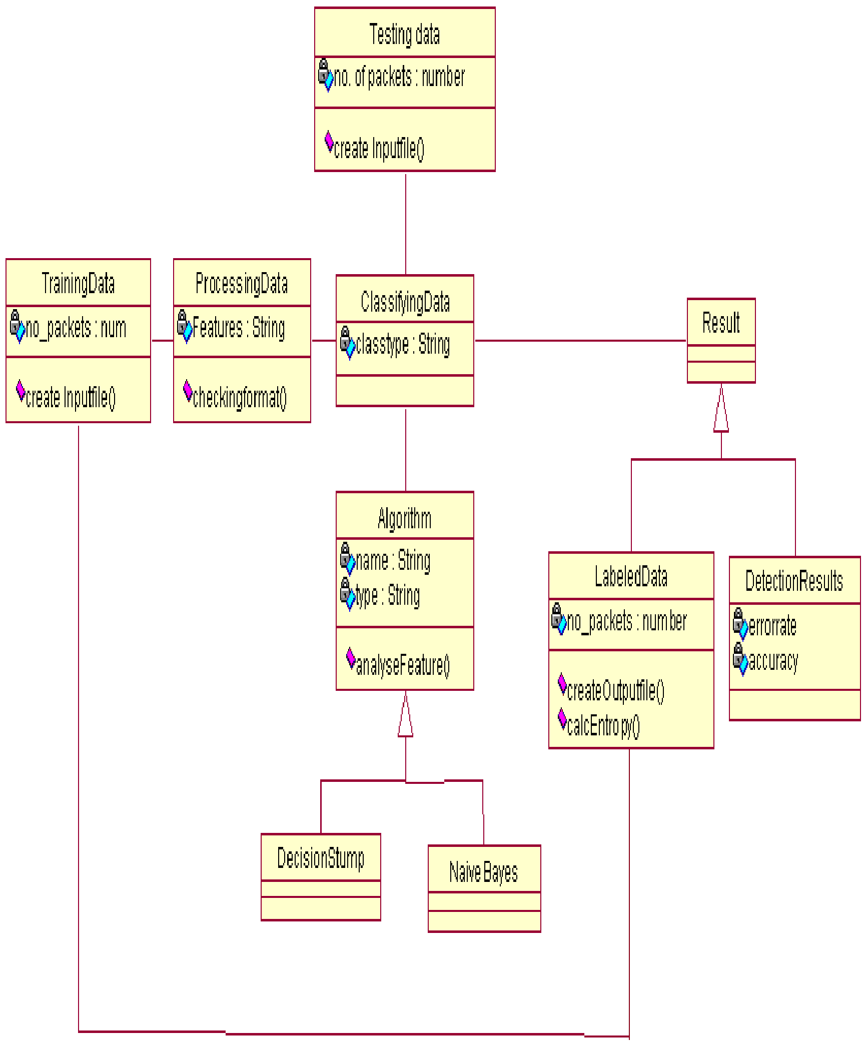


Fig. 3.4 Class diagram

3.5.3 State transition Diagram

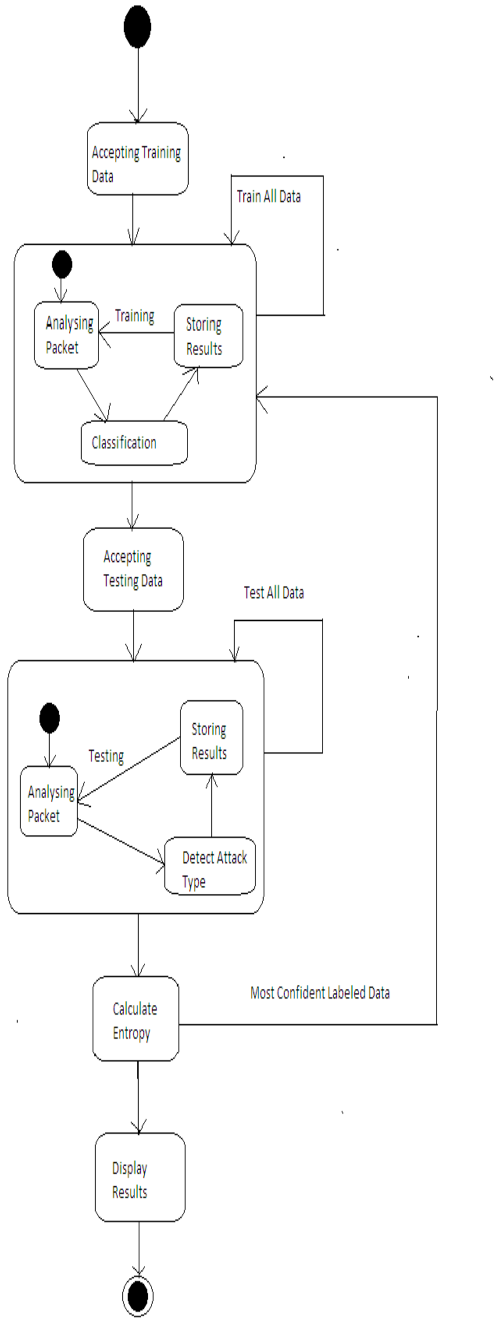


Fig.3.5 State transition diagram

CHAPTER 4

SEMI-SUPERVISED APPROACH

Semi-supervised learning is a class of supervised learning tasks and techniques that also make use of unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent (e.g. to transcribe an audio segment) or a physical experiment (e.g. determining the 3D structure of a protein or determining whether there is oil at a particular location). The cost associated with the labeling process thus may render a fully labeled training set infeasible, whereas acquisition of unlabeled data is relatively inexpensive. In such situations, semi-supervised learning can be of great practical value.

Semi-supervised learning is also of theoretical interest in machine learning and as a model for human learning. Semi-supervised learning attempts to make use of this combined information to surpass the classification performance that could be obtained either by discarding the unlabeled data and doing supervised learning or by discarding the labels and doing unsupervised learning. Semi-supervised learning may refer to either transductive learning or inductive learning. The goal of transductive learning is to infer the correct labels for the given unlabeled data. The goal of inductive learning is to infer the correct mapping. Intuitively, we can think of the learning problem as an exam and labeled data as the few example problems that the teacher solved in class. The teacher also provides a set of unsolved problems. In the transductive setting, these unsolved problems are a take-home exam and you want to do well on them in particular. In the inductive setting, these are practice problems of the sort you will encounter on the in-class exam. It is unnecessary to perform transductive learning by way of inferring a classification rule over the entire input space; however, in practice, algorithms formally designed for transduction or induction are often used interchangeably.

In order to make any use of unlabeled data, we must assume some structure to the underlying distribution of data. Semi-supervised learning algorithms make use of at least one of the following assumptions.

a. Smoothness assumption: Points that are close to each other are more likely to share a label. This is also generally assumed in supervised learning and yields a preference for geometrically simple decision boundaries. In the case of semi-supervised learning, the smoothness assumption additionally yields a preference for decision boundaries in low-density regions, so that there are fewer points close to each other but in different classes.

b. Cluster assumption: The data tend to form discrete clusters, and points in the same cluster are more likely to share a label (although data sharing a label may be spread across multiple clusters). This is a special case of the smoothness assumption.

c. Manifold assumption: The data lie approximately on a manifold of much lower dimension than the input space. In this case we can attempt to learn the manifold using both the labeled and unlabeled data to avoid the curse of dimensionality. Then learning can proceed using distances and densities defined on the manifold. The manifold assumption is practical when some process that may be hard to model directly, but which only has a few degrees of freedom is generating high-dimensional data. For instance, speech output is controlled by a series of vocal tubes, and a few muscles control images of various facial expressions. We would like in these cases to use distances and smoothness in the natural space of the generating problem, rather than in the space of all possible acoustic waves or images respectively.

The heuristic approach of self-training (also known as self-learning or self-labeling) is historically the oldest approach to semi-supervised learning, with examples of applications starting in the 1960s. Vladimir Vapnik formally introduced the transductive learning framework in the 1970s. Interest in inductive learning using generative models also began in the 1970s. A probably approximately correct learning bound for semi-supervised learning of a Gaussian mixture was demonstrated by Ratsaby and Venkatesh in 1995. Semi-supervised learning has recently become more popular and practically relevant due to the variety of problems for which vast quantities of unlabeled data are available—e.g. text on websites, protein sequences, or images.

CHAPTER 5

SYSTEM DESIGN

5.1 System Architecture

The intrusion detection system can analyze the dataset packets from the database and detect whether it would be an intrusion or not. From the view of architecture, the diagram of system includes several modules, which has shown in Figure 1.

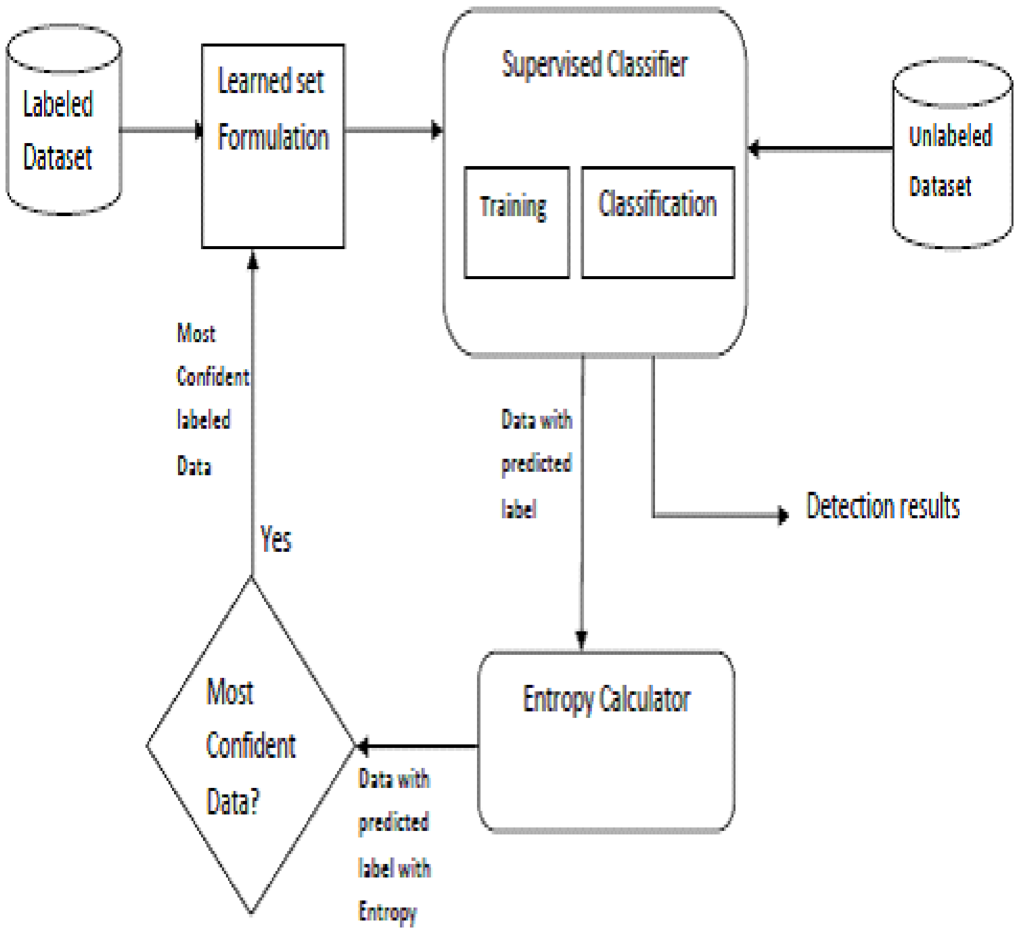


Fig.5.1 Architecture for semi-supervised IDS using self-learning algorithm (SLA)

Based on the figure above, there are several modules that introduce the intrusion detection system architecture. The following explains the layering structure step by step:

- **Labeled Dataset:** This dataset is the training set which has labels and is taken from KDD99 dataset. The labeled dataset should contain only small number of packets and has to be in correct format. This dataset needs to be browsed by the user when performing the operation.
- **Learned Set Formulation**: This module consists of training data as well as data which has been correctly classified from the testing set. This results in increase of training data which will thus result in giving more accurate results.
- **Unlabeled Dataset**: This dataset is the testing set which does not have labels and is taken from KDD99 dataset. The unlabeled dataset should contain only small number of packets and has to be in correct format. This dataset needs to be browsed by the user when performing the operation.
- **Supervised Classifier**: The actual algorithm will function in this module. This module takes as input the learned set formulation set and unlabeled set. Training is performed then classification takes place.
- **Detection Results**: These results include accuracy rate, detection rate, etc. as well as whether the packet is a normal packet or attack packet.
- **Entropy Calculator**: After application of Supervised Classifier, the testing data will have class labels in it. These packets now need to be checked whether they are correctly classified. Thus their entropy is calculated and the packets having entropy above some threshold are said to be correctly classified.
- **Most Confident Data**: These are the packets which have entropy above some threshold and are said to be most confident. These packets are then added in the learned set formulation set.

All of these modules together make the IDS system architecture based on the Machine learning techniques, and the function of each module has been introduced briefly. The present study is aimed to study the algorithm that can prove best to reduce the false alarm rate and increase the detection rate that also using semi-supervised learning.

5.2 UML Diagrams

5.2.1 Use case Diagram

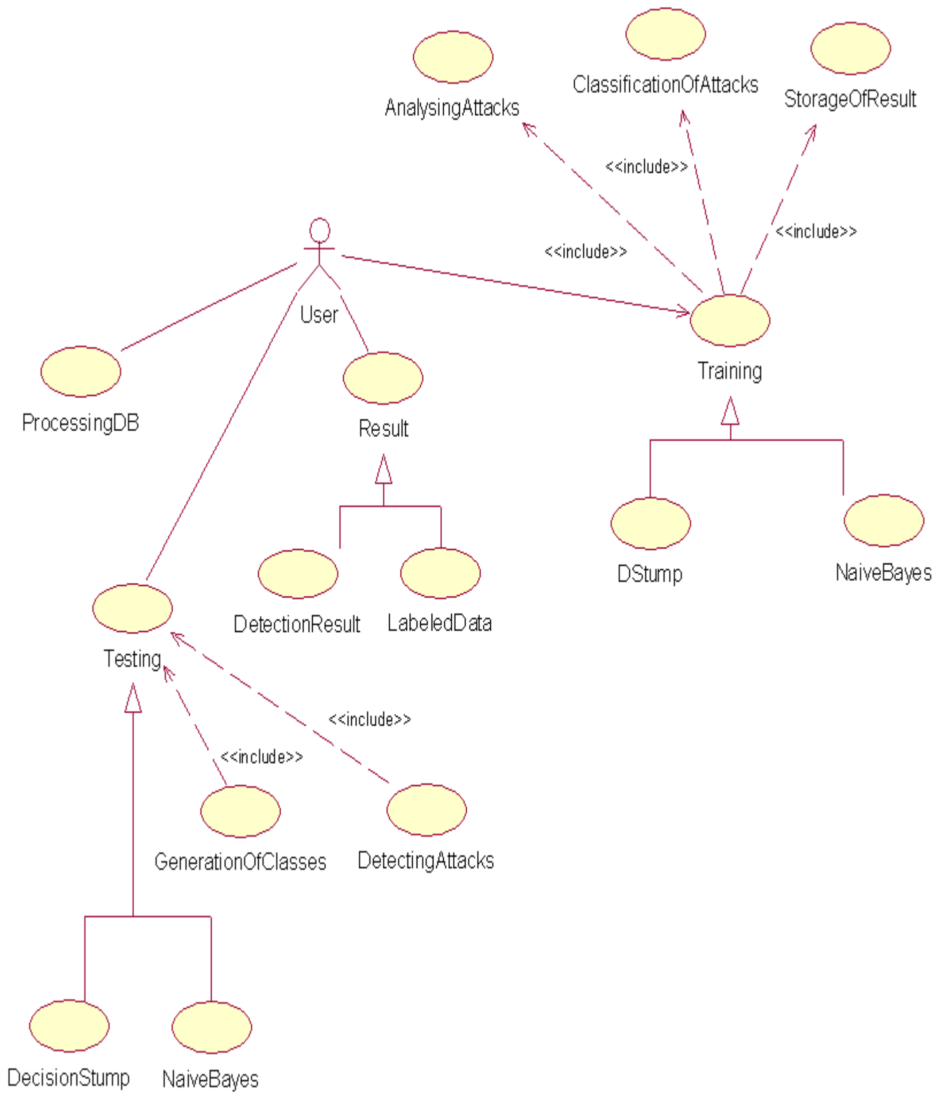


Fig.5.2 Use case Diagram

5.2.2 Activity Diagram

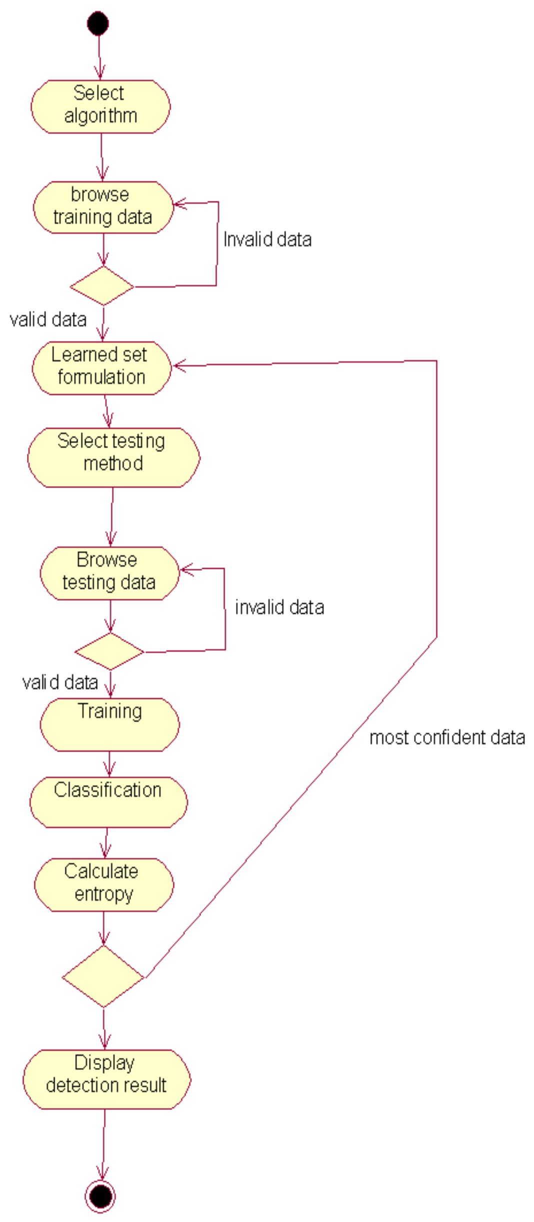


Fig.5.3 Activity Diagram

5.2.3 Sequence Diagram

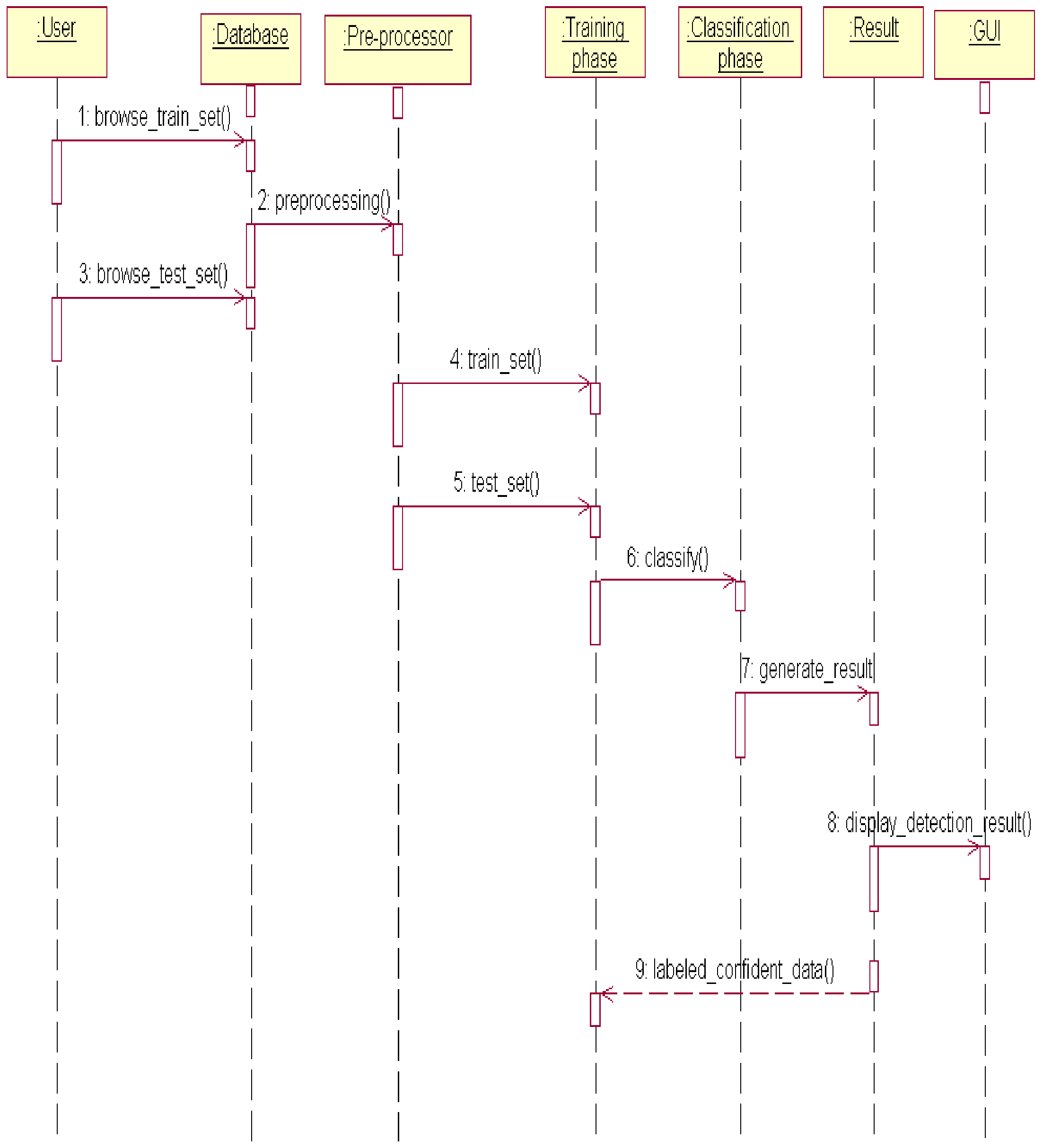


Fig.5.4 Sequence Diagram

CHAPTER 6

TECHNICAL SPECIFICATION

6.1 Domain Area of Project

The domain of the project is Machine Learning and network security.

6.2 Technology Consideration

- Windows 7
- Eclipse Indigo
- Netbeans IDE

6.2.1 Features of Java

➢ **Simple**

- Looks familiar to existing programmers: related to C and C++:
- Omits many rarely used, poorly understood, confusing features of C++, like operator overloading, multiple inheritance, automatic coercions, etc.
- Contains no *goto* statement, but *break* and *continue*
- Has no header files and eliminated C pre-processor
- Eliminates much redundancy (e.g. no structs, unions, or functions)
- Has no pointers
- Garbage collection, so the programmer won't have to worry about storage management, which leads to fewer bugs.
- A rich predefined class library

➢ **Object-Oriented**

Java is an object-oriented language, which means that you focus on the *data* in your application and *methods* that manipulate that data, rather than thinking strictly in terms of procedures. In an object-oriented system, a *class* is a collection of data and methods that operate on that data. Taken together, the data and methods describe the state and behavior of an *object*. Classes are arranged in a hierarchy, so that a subclass can inherit behavior from its superclass. Java comes with an extensive set of classes, arranged in *packages*, that you can use in your programs.

- **Distributed**

  - It has a spring-like transparent RPC system
  - Now uses mostly tcp-ip based protocols like ftp & http

Java supports various levels of network connectivity through classes in the java.net package (e.g. the URL class allows a Java application to open and access remote objects on the internet).

- **Interpreted**

The Java compiler generates *byte-codes*, rather than native machine code. To actually run a Java program, you use the Java interpreter to execute the compiled byte-codes. Java byte-codes provide an architecture-neutral object file format. The code is designed to transport programs efficiently to multiple platforms.

- rapid turn-around development
- Software author is protected, since binary byte streams are downloaded and not the source code

- **Robust**

Java has been designed for writing highly reliable or robust software:

- language restrictions (e.g. no pointer arithmetic and real arrays) to make it impossible for applications to smash memory (e.g overwriting memory and corrupting data)
- Java does automatic garbage collection, which prevents memory leaks
- extensive compile-time checking so bugs can be found early; this is repeated at runtime for flexibilty and to check consistency

➢ **Secure**

Security is an important concern, since Java is meant to be used in networked environments. Without some assurance of security, you certainly wouldn't want to download an applet from a random site on the net and let it run on your computer. Java's memory allocation model is one of its main defenses against malicious code (e.g can't cast integers to pointers, so can't forge access). Furthermore:

- access restrictions are enforced (public, private)
- byte codes are verified, which copes with the threat of a hostile compiler

➢ **Architecture-Neutral**

- compiler generates byte codes, which have nothing to do with a particular computer architecture
- easy to interpret on any machine

➢ **Portable**

Java goes further than just being architecture-neutral:

- no "implementation dependent" notes in the spec (arithmetic and evaluation order)
- standard libraries hide system differences
- the Java environment itself is also portable: the portability boundary is POSIX compliant

➢ **High-Performance**

Java is an interpreted language, so it will never be as fast as a compiled language as C or C++. In fact, it is about 20 times as slow as C. However, this speed is more than enough to run interactive, GUI and network-based applications, where the application is often idle, waiting for the user to do something, or waiting for data from the network.

- **Multithreaded**

Java allows multiple concurrent threads of execution to be active at once. This means that you could be listening to an audio clip while scrolling the page and in the background downloading an image. Java contains sophisticated synchronization primitives (monitors and condition variables), that are integrated into the language to make them easy to use and robust. The *java.lang* package provides a *Thread* class that supports methods to start, run, and stop a thread, and check on its status.

- **Dynamic**

Java was designed to adapt to an evolving environment:

- Even after binaries have been released, they can adapt to a changing environment
- Java loads in classes as they are needed, even from across the network
- It defers many decisions (like object layout) to runtime, which solves many of the version problems that C++ has
- Dynamic linking is the only kind that exists

6.2.2 History of Java

James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time. The language was initially called Oak after an oak tree that stood outside Gosling's office; it went by the name Green later, and was later renamed Java, from Java coffee said to be consumed in large quantities by the language's creators. Gosling aimed to implement a virtual machine and a language that had a familiar C/C++ style of notation.

Sun Microsystems released the first public implementation as Java 1.0 in 1995. It promised "Write Once, Run Anywhere" (WORA), providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run Java applets within web pages, and Java quickly became popular. With the advent of Java

2(released initially as J2SE 1.2 in December 1998 – 1999), new versions had multiple configurations built for different types of platforms. For example, J2EE targeted enterprise applications and the greatly stripped-down version J2ME for mobile applications (Mobile Java). J2SE designated the Standard Edition. In 2006, for marketing purposes, Sun renamed new J2 versions as Java EE, Java ME, and Java SE, respectively.

In 1997, Sun Microsystems approached the ISO/IEC JTC1 standards body and later the Ecma International to formalize Java, but it soon withdrew from the process. Java remains a de facto standard, controlled through the Java Community Process. At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System. Sun distinguishes between its Software Development Kit (SDK) and Runtime Environment (JRE) (a subset of the SDK); the primary distinction involves the JRE's lack of the compiler, utility programs, and header files.

On November 13, 2006, Sun released much of Java as free and open source software, (FOSS), under the terms of the GNU General Public License (GPL). On May 8, 2007, Sun finished the process, making all of Java's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.

Sun's vice-president Rich Green said that Sun's ideal role with regards to Java was as an "evangelist" Following Oracle Corporation's acquisition of Sun Microsystems in 2009–2010, Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency". This did not hold Oracle, however, from filing a lawsuit against Google shortly after that for using Java inside the Android SDK. Java software runs on everything from laptops to data centers, game consoles to scientific supercomputers. There are 930 million Java Runtime Environment downloads each year and 3 billion mobile phones run Java. On April 2, 2010, James Gosling resigned from Oracle.

CHAPTER 7

PROJECT ESTIMATE, SCHEDULE AND TEAM STRUCTURE

7.1 Feasibility Study

The feasibility study of our project "Pattern Based Security using machine learning techniques" involves the following constraints:

7.1.1 Technical Feasibility

The technology and the resources used to develop the system involve the following components.

1. Hardware Components: Computer system

2. Software Components: Eclipse, Netbeans IDE, Windows 7

The technical specifications of the system imply that the system is very well adaptable to the available technology and does not demand any more technically advanced, unavailable or infeasible equipment.

7.1.2 Economical Feasibility

The resources required by the system are available easily in the market at reasonable cost. The various costs incurred in the system are as given below:

Table 7.1 Component Cost

Component	Cost
Personal Computer (Intel Core2Duo CPU 2.5 GHz , 1 GB RAM)	Rs. 25,000.00
Windows Operating System	Rs. 6,490.00
Eclipse	Rs. 0

Viewing above specifications, the system was found to be economically feasible for the concerned user. The above specified are onetime costs.

7.1.3 Time Feasibility

There were several studies carried out during the project development. This task was done successfully. Updates were kept, the weekly report of advancement during project time. Various issues and ideas are discussed in the meeting with the College Staff and the Project Guide and we strive to get better solutions.

7.1.4 Operational Feasibility

Modern PCs are becoming easier to handle. Also the system is providing both types of input i.e. keyboard and mouse. The users who are comfortable with using PCs find this application operationally much more feasible.

7.2 Resources

7.2.1 Human Resources

The total human resources required for the development of the project is four. The testing resources required to test the project is four.

7.2.2 System Requirements

1. Hardware Requirements

"Pattern Based Security Using Machine Learning Techniques" for windows has following minimum hardware requirements.

- 2.4 GHz CPU
- 2GBRAM
- 80 GB Hard Drive
- Keyboard, mouse(optical mouse controls work the best)
- Colour Monitor screen resolution 1024x768

2. Software Requirements

"Pattern Based Security Using Machine Learning Techniques" has the following software requirements.

- Windows 7
- Eclipse

7.3 Scheduling

Table 7.2 Project Duration

Project Start Date	26-July-2014
Project End Date	21-April-2014
Project Duration	9 months

Table 7.3 Phases of Project

Sr. No.	Milestone Name	Milestone Description	Timeline	Remarks
1	Design of Project Overview Construction	Information relating to the deliverable at this milestone	11	15%
2	Developing and testing of modules	Shows the participants in the project and an overall design.	16	50%
3	Final Integration and Project testing	Preparing and designing the User Interface, Command, Syntax and Semantic Recognition, Command Implementation, and additional utilities	2	30%
4	Documentation and Project report and Presentation	Testing of project and specification	2	5%

7.4 Risk Management

Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project. A risk is a potential problem-it might happen it might not. But, regardless of the outcome, it's a really good idea to identify it, access its probability of occurrences, estimate its impact, and establish a contingency plan should the problem actually occur.

For any project developed, it is empirical for the team to monitor and manage risks. Risks are the unexpected delays and hindrances that are faced by the software and need to be actively managed for rapid development. In the words of Tom Glib, "if you don't actively attack risks, they will actively attack you."

The key functions of software risk management are to identify, address and eliminate sources of risk before they become threats to successful completion of a software project. An effective strategy must address three issues:

- Risk avoidance
- Risk monitoring
- Risk management and contingency planning

If a software team adopts a proactive approach to risk, avoidance is the best strategy. This is achieved by developing a plan for risk mitigation. As the project proceeds, risk-monitoring activities commence and the project manager monitors factors that may provide indication of whether the risk is becoming more or less likely. Risk management and contingency planning assumes that mitigation efforts have failed and the risk has become a reality. The RMMM plan tackles risks through risk assessment and risk control. Risk assessment involves risk identification, risk analysis and risk prioritization; while risk control involves risk management planning, risk resolution and risk monitoring.

7.4.1 Risk Table

The risks are categorized on the basis of their occurrence and the impact that would have, if they do occur. Their impact is rated as follows:

1. Catastrophic

2. Critical

3. Marginal

4. Negligible

Table 7.4 Risk Table

Risk	**Category**	**Probability**	**Impact**
1. Size estimate may be low	Project	50%	Marginal
2. Stricter completion Schedule	Business	10%	Critical
3. Sophistication of the end users application program	Business	40%	Marginal
4. Less reuse than planned	Technical	25%	Marginal
5.Poor Quality documentation	Business	30%	Critical
6. Debugging phase may take more time than expected	Technical	30%	Marginal
7. Poor comments in the code	Technical	20%	Negligible
8. Lack of technical support on unforeseen environment	Technical	35%	Critical
9. Increase on the workload on the developers and hence divided attention on project	Personal	40%	Critical
10. Schedule might slip due to inexperienced persons	Personal	40%	Critical
11. Hardware Risks	Support	40%	Critical
12. Environmental Risks	Performance	20%	Marginal
13. Security Risks	Security	20%	Catastrophic

7.4.2 RMMM (Risk Mitigation, Monitoring and Management) Plan

- **Risk 1:**
  Size estimate may be high.
  Mitigation: keep on modularizing the software and estimate size of individual modules.
  Monitor: module to be written.
  Management: increase members in the more time consuming modules.

- **Risk 2:**
  Strict completion schedule.
  Mitigation: schedule the project in such a fashion that major modules are completed first so that they get enough time for testing and debugging.
  Monitor: keep track of the schedule slips.
  Management: put in extra hours to make up for the lost hours.

- **Risk 3:**
  Less reuse than planned.
  Mitigation: use OOPS concepts such as classes, objects, functions etc.
  Monitor: modules should not be written containing similar functionality
  Management: revise all the modules and take necessary steps to make it reusable.

- **Risk 4:**
  Sophistication of the end users application program.
  Mitigation: complex queries and function are to be implemented.
  Monitor: every minor issue relating the query is concerned
  Management: customer should be clearly notified about the limitations of the functionalities.

- **Risk 5:**
  Debugging phase may take more time than expected.
  Mitigation: use debugging tools if available.
  Monitor: Ensure that debugging tools are properly used or not.
  Management: increase the no of hours to be worked for each member.

- **Risk 6:**
  Poor comments in the code.
  Mitigation: commenting standards are better studied and expressed
  Monitor: review of the code with special attention given to comments will determine if they are up to standard.
  Management: time must be made available to make comments up to the standards.

7.5 Team Structure

Sr. No.	Work	Team Members
1.	Decision Tree Algorithm	Anagha Khati, Auzita Irani, Naba Inamdar, Rashmi Soni
2.	Testing	Anagha Khati, Naba Inamdar
3.	Entropy Module	Anagha Khati, Naba Inamdar
4.	Semi-supervised Module	Naba Inamdar
5.	Result Analysis	Anagha Khati, Auzita Irani, Naba Inamdar, Rashmi Soni
6.	Graphical User Interface	Naba Inamdar, Rashmi Soni
7.	Documentation	Anagha Khati, Auzita Irani, Naba Inamdar, Rashmi Soni

CHAPTER 8

SOFTWARE IMPLEMETATION

8.1 Introduction

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. We implemented our project through machine learning techniques. Machine learning algorithms used are J48, Naïve Bayes. During implementation proposed speed of execution of modules, handling database and data structure is a tough task. Selecting proper data structure is very important for executing programs. Choosing the most appropriate language for implementation, whether it is platform dependent or not and handling all various types of files for execution should be kept in mind. Now are project deals is done in JAVA which is OS independent and pure object oriented.

8.2 Database

KDD dataset is Knowledge Discovery Database. KDD 99 is a standard source of data for evaluating IDSs which appeared in 1990 which is used in our project. The data includes the US air force local network together with a variety of simulated attacks. In this collection, every sample is equal to a connection. A connection is a sequence of TCP packages which starts and ends at a specific time with the flow of data form source IP address to the destination IP. 41 features were defined for each connection in this collection, which are divided into four major categories namely: primary features of TCP protocol, content features, time–based and host-based traffic features. Every connection has a label which determines whether it is normal or it is one of the defined attacks. The present attacks are divided into four categories as follows: U2R, R2L, DOS and PROBE. Note that NORMAL is a normal connection. 53

8.3 Important Modules and Algorithms

8.3.1 J48 Algorithm

The J48 Decision tree classifier follows the following simple algorithm. In order to classify a new item, it first needs to create a decision tree based on the attribute values of the available training data. So, whenever it encounters a set of items (training set) it identifies the attribute that discriminates the various instances most clearly. This feature that is able to tell us most about the data instances so that we can classify them the best is said to have the highest information gain. Now, among the possible values of this feature, if there is any value for which there is no ambiguity, that is, for which the data instances falling within its category have the same value for the target variable, then we terminate that branch and assign to it the target value that we have obtained.

For the other cases, we then look for another attribute that gives us the highest information gain. Hence we continue in this manner until we either get a clear decision of what combination of attributes gives us a particular target value, or we run out of attributes. In the event that we run out of attributes, or if we cannot get an unambiguous result from the available information, we assign this branch a target value that the majority of the items under this branch possess. Now that we have the decision tree, we follow the order of attribute selection as we have obtained for the tree. By checking all the respective attributes and their values with those seen in the decision tree model, we can assign or predict the target value of this new instance. The J48 Decision tree classifier follows the following s algorithm:

Generate_decision_tree: Generates a decision tree from the training tuples of data partition D.

Input:

1. D is a set of training tuples
2. Attribute list
3. Attribute selection method

Output: A Decision tree

Method

```
1    create a node N;
2    if tuple in D are all of the same class, C, then
3    return N as a leaf node labeled with the class C;
4    if attribute_list is empty then
5    return N as a leaf node labeled with majority class in D;
6    apply Attribute_selection_method(D,attribute_list) to find the
     "best" splitting_criterion;
7    label node N with splitting_criterion;
8    if splitting_attribute is discrete-valued and multiway splits allowed
     then
9    attribute_list <- attribute_list – splitting_attribute;
10   for each outcome j of splitting_criterion
11   let Dj be the set of data tuples in D satisfying outcome j;
12   if Dj is empty then
13   attach a leaf labeled with the majority class in D to node N;
15   return N;
```

8.3.1.1 Modules

For implementation of J48 algorithm we create following modules:

8.3.2 Naive Bayes Algorithm

The Naive Bayes classifier works on a simple, but comparatively intuitive concept. Also, in some cases it is also seen that Naive Bayes outperforms many other comparatively complex algorithms. It makes use of the variables contained in the data sample, by observing them individually, independent of each other. The Naïve Bayes classifier is based on the Bayes rule of conditional probability. It makes use of all the attributes contained in the data, and analyses them individually as though they are equally important and independent of each other.

8.3.2.1 Training Algorithm

Steps No.	**Steps**
1	Read training data packets line by line till end of file
1.1	Copy entire packet in dictionary and neglect the duplicates of every attribute of each packet
1.2	Copy attack name as well and keep count of attacks identified
2	Create a table which will contain all unique attribute values of all 41 features
3	Insert number of times an attribute value appearing for every attack recognized during training in the table in a list where attribute values are stored.
4	Count number of packets in training.
5	Count number of times each minor attack encountered during training and update all major attacks list after identifying its major attack.

8.4 Business Logic and Architecture

8.4.1 Business Logic

The work which is being done and implemented needs to be documented for the ones who wants to use in their research and development, case studies and various subjects and plenty more wants to know what has been done till now in a specified work, so that what advance can be done to make to it creative and what different can be presented to the people to make it innovative which is nothing but new change and turnaround in the field of technology and how this entire package can boost to people to their new ideas in the same field. Especially to the researchers, developers, organization, company, etc. After a brief study of various papers studying such as "Improving the Accuracy of Intrusion Detection Systems by using the Combination of Machine Learning Approaches", **"**A Comprehensive Analysis and study in Intrusion Detection System using

Data Mining Techniques", "A survey on intrusion detection techniques", we also made a paper in the IEEE format which was selected in National Conference "NCSEE 2014, Pune" and had our paper published in the Journal of Harmonized Research.

Our project is built and developed using three algorithms named J48, Naive Bayes. Working of J48 independently and implementing in the domain Network Security with various approaches and getting good and accurate results is tremendous contribution in the Network environment if we publish the paper. Paper will be helpful to many organizations, viewers and the ones who are working in the network security environment to add a new document to their study. Naive Bayes can also be considered foe same matter. This was regarding only 2 algorithms of our project can be used as a reference for further development in field of Network Security.

Apart from Algorithms many more topics are covered in our project which can be called as entire package containing more useful contents like Mathematical Model, UML & DFD Diagrams, SRS, Testing Plans, and Scheduling. Each of them as a unit provides a guidelines to the ones who needs example for their case study. Mathematical model used in mathematics and models created by us can be used can be useful in showing example with reference to the context. Diagrams are very good understandable which is actually visualization of entire project but, it starts from small modules which help in studying how to draw UML diagrams. All the other things mentioned in the list can also be helpful in similar way.

8.4.2 Architecture

Our project name is "Pattern based security using Machine Learning Techniques". Intrusion is nothing but disturbance or an obstacle one. An intrusion detection system (IDS) is a system for the detection of intrusions. Intrusion detection involves detecting unusual patterns of activity or patterns of activity that are known to correlate with intrusions. Machine Learning means basically giving intelligence to the machine to learn itself just like how the kid starts learning by himself through teachings from his parents similarly machines learn like that. A program is written so that it keeps learning and shows the end result satisfying the developer. Now shifting to main project idea it purely deals with all about detecting attacks happened on the data packets on the network platform and to reduce false alarm rate. FAR is False Alarm Rate which means indicating a wrong signal for correct one and correct signal for the wrong one. To implement this project we are using KDD dataset which is Knowledge Discovery Database. KDD 99 is a standard source of data for evaluating IDSs which appeared in 1990. The data includes the US air force local network together with a variety of simulated attacks. In this collection, every sample is equal to a connection. A connection is a sequence of TCP packages which starts and ends at a specific time with the flow of data form source IP address to the destination IP. 41 features were defined for each connection in this collection, which are divided into four major categories namely: primary features of TCP protocol, content features, time–based and host-based traffic features. Every connection has a label which determines whether it is normal or it is one of the defined attacks. The present attacks are divided into four categories as follows: U2R, R2L, DOS and PROBE. Note that NORMAL is a normal connection.

Two algorithms are being used for identifying major attacks DOS, Probe U2r, R2l and normal. J48, Naive Bayes are the two machine learning algorithm used for detection. Considering 10% of KDD dataset 40% of data are used for training and 60% for testing. Now training actually means learning, basically each time data is trained through different algorithm program comes to know attributes values & its attribute attack type. So basically machine becomes intelligent to identify and recognize attack during testing phase. While training machine knows, if during testing this packets comes it becomes easy for machine to identify the packet and display attack type. If packets other than packets used for training appears then it uses its intelligence which was build up during and training phase to recognize attack type. So input to the project is KDD Dataset and machine learning algorithms are used for training and testing. First data is trained and then testing is done to check the results and our motive is to reduce False Alarm Rate.

CHAPTER 9

SOFTWARE TESTING

9.1 Introduction

Software Testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Software is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails. Most physical systems fail in a fixed (and reasonably small) set of ways. By contrast, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible.

9.2 Test Specification

9.2.1 Introduction

Software Testing can be defined as: Testing is an activity that helps in finding out bugs/defects/errors in a software system under development, in order to provide a bug free and reliable system/solution.

9.2.2 Test Plan

Table 9.1: Test Cases

Case ID	**Test Case Name**	**Inputs**	**Expected Output**	**Actual Output**	**Remark**
1	Input training data	KDDCup dataset	Correct data	Correct data	Yes
2	Input training data	KDDCup dataset	Correct data	Incorrect Feature Values	No
3	Input training data	KDDCup dataset	Correct data	Dataset not in correct	No

				format	
4	Input training data	Other than KDDCup dataset	Correct data	Wrong dataset	No
5	Input testing data	KDDCup dataset	Correct data	Correct data	Yes
6	Input testing data	KDDCup dataset	Correct data	Incorrect Feature Values	No
7	Input testing data	KDDCup dataset	Correct data	Dataset not in correct format	No
8	Input testing data	Other than KDDCup dataset	Correct data	Wrong dataset	No

9.2.3 Functional Testing

Functional testing is performed on the project management software. The functional testing allows us to test the various functionality of the software. Black box testing takes an external perspective of the test object to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid input and determines the correct output. There is no knowledge of the test object's internal structure. The software is tested manually.

9.2.4 Function Point Analysis

Function point analysis for project management software which provides a mechanism that both software developers and users could utilize to define functional requirements. It determines a best way to gain an understanding of the user's needs.

- **Data Functions**

1. **Internal Logical files**

The internal logical files allow the user to maintain the data in the database that is provided by the user from a user interface of from the external by system (ILF).

2. **External Interface files**

The External Interface files are one in the data will be stored in the other system and the user is not responsible for maintaining the data in the database (EIF).

- **Transactional Functions**

1. **External Inputs**

External Input is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application (EI).

2. **External Outputs**

External Output is an elementary process in which derived data passes across the boundary from inside to outside (EO).

3. **External Inquiries**

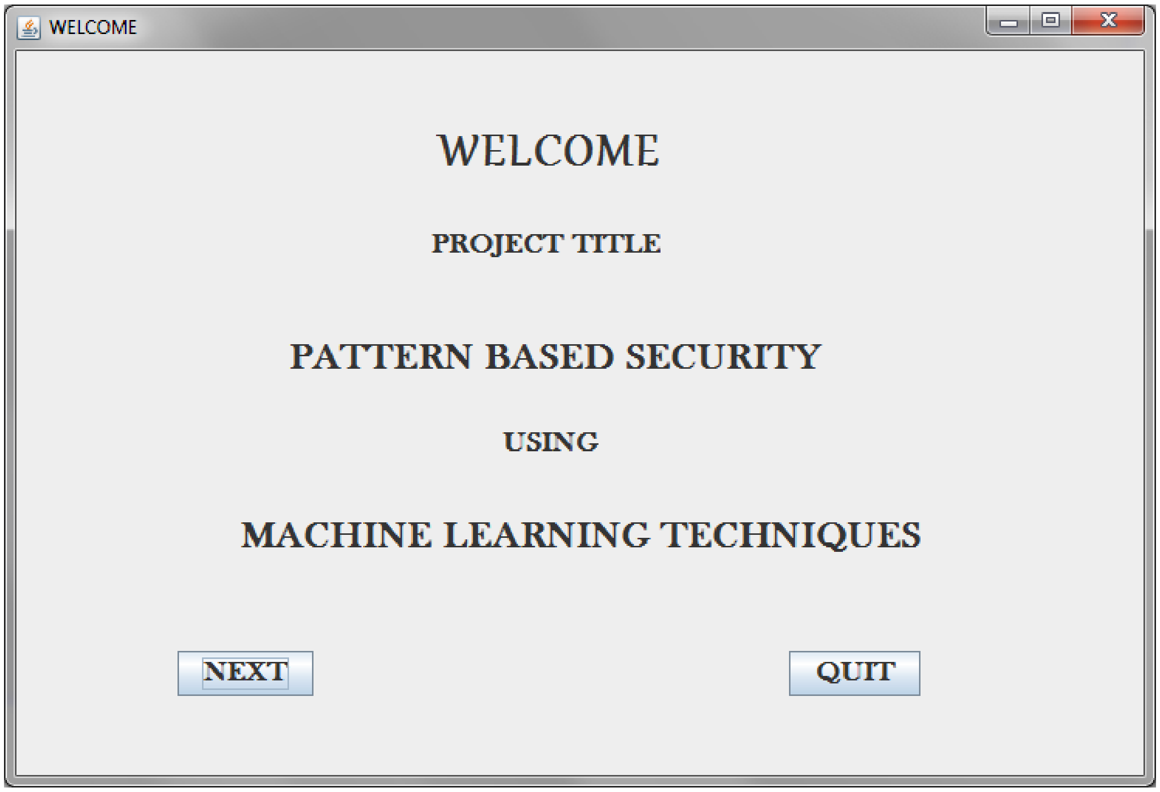
External Inquiries is an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files (EQ).

CHAPTER 10

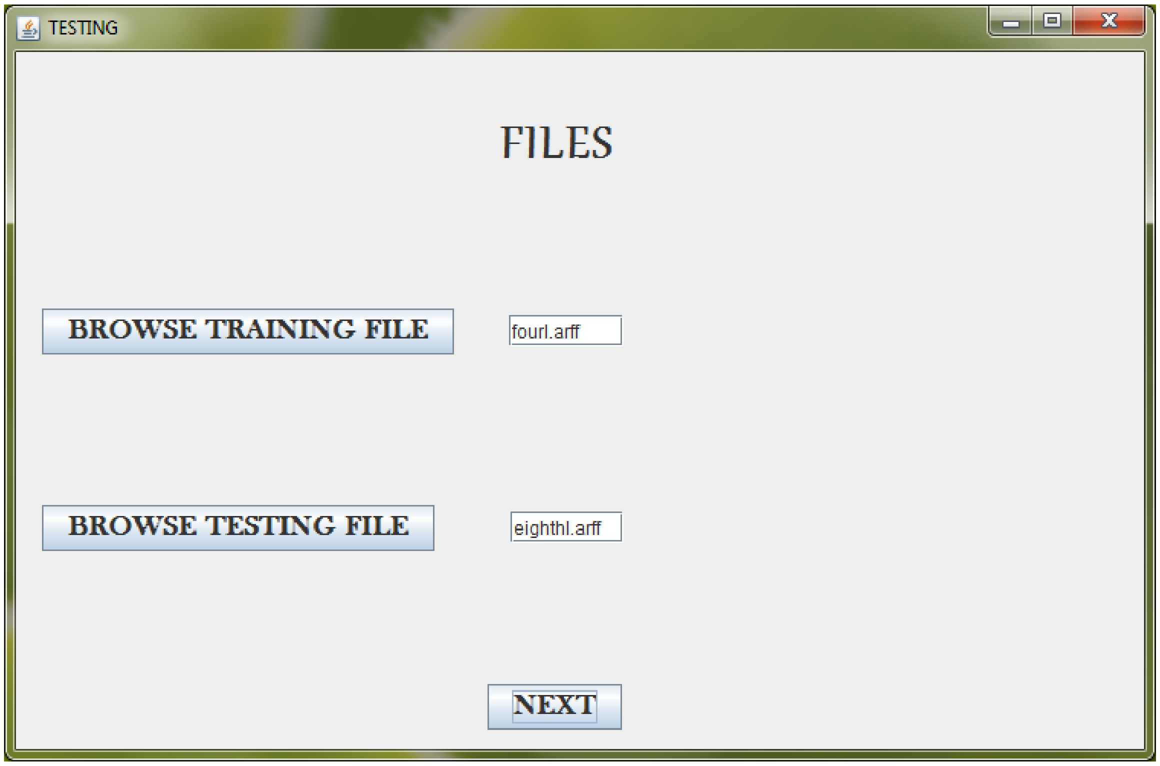
RESULTS

10.1 GUI Snapshots

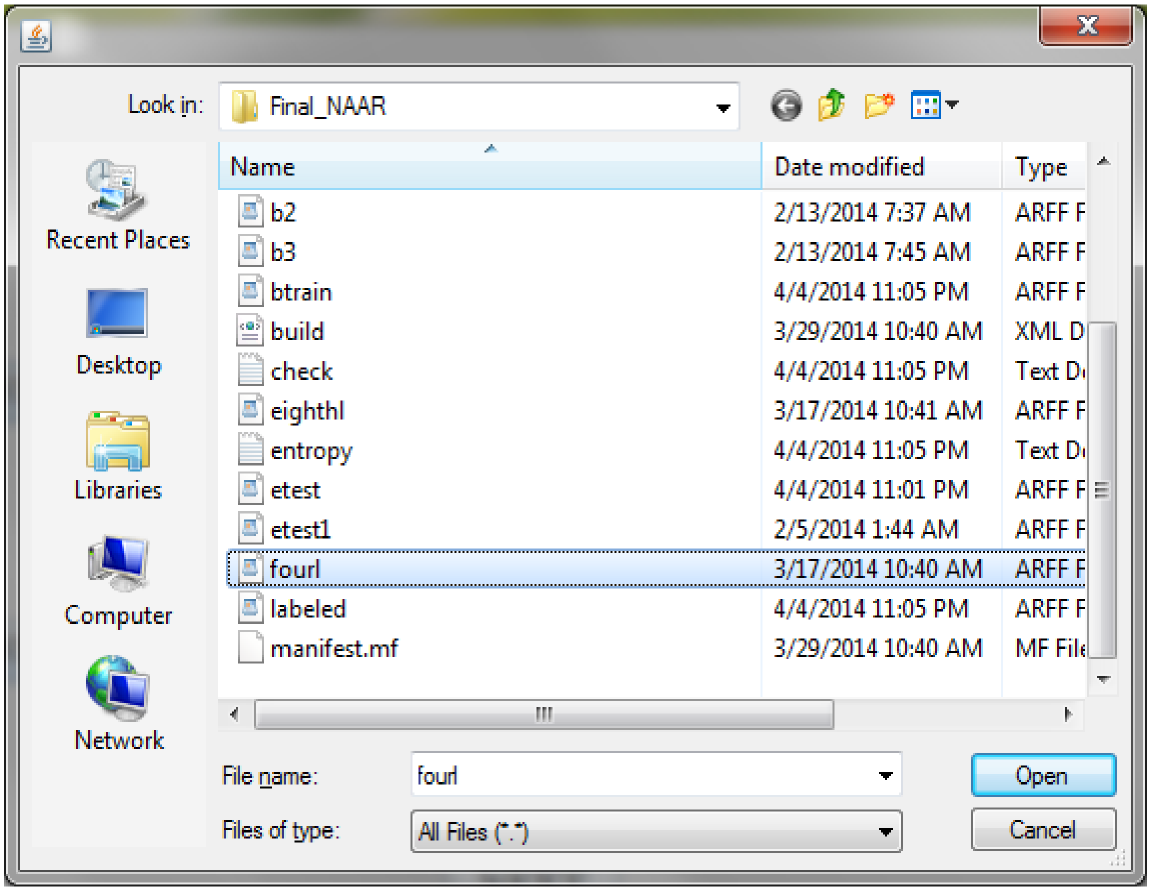
1. Welcome Screen:



2. Training/Testing File Selection:



3. Browse KDD Data Set:



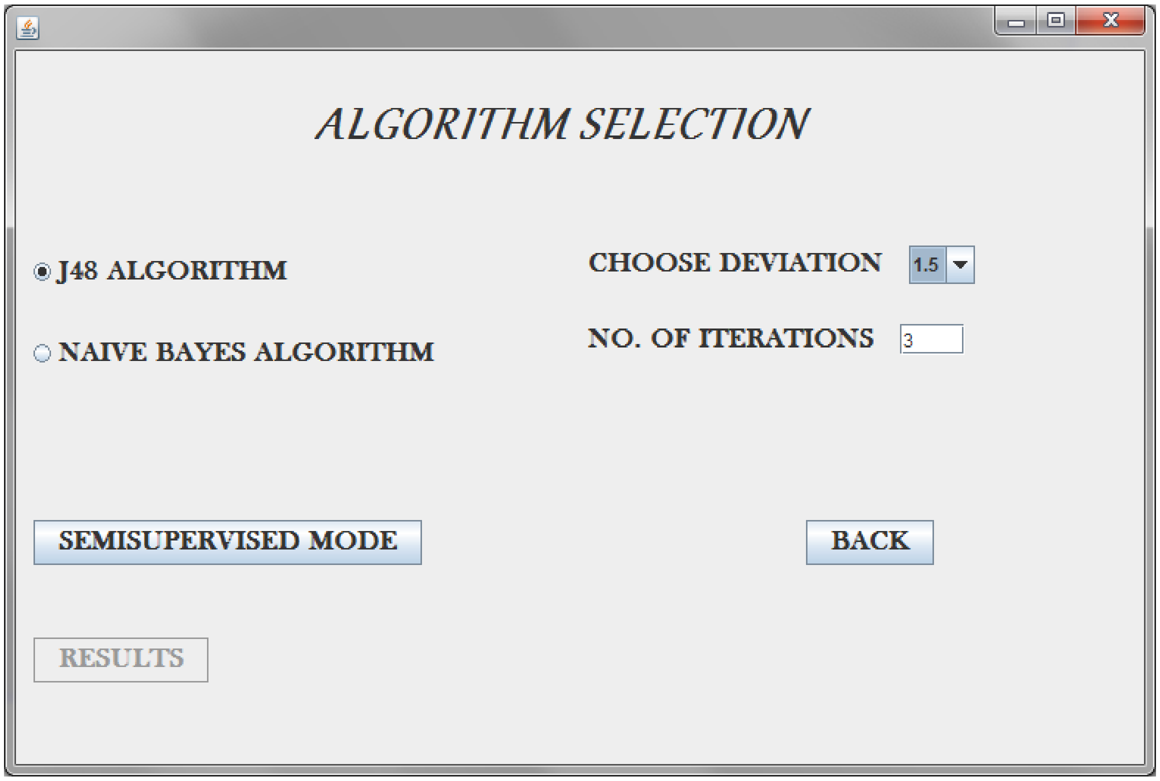
Look in: Final_NAAR

Name	Date modified	Type
b2	2/13/2014 7:37 AM	ARFF F
b3	2/13/2014 7:45 AM	ARFF F
btrain	4/4/2014 11:05 PM	ARFF F
build	3/29/2014 10:40 AM	XML D
check	4/4/2014 11:05 PM	Text D
eighthl	3/17/2014 10:41 AM	ARFF F
entropy	4/4/2014 11:05 PM	Text D
etest	4/4/2014 11:01 PM	ARFF F
etest1	2/5/2014 1:44 AM	ARFF F
fourl	3/17/2014 10:40 AM	ARFF F
labeled	4/4/2014 11:05 PM	ARFF F
manifest.mf	3/29/2014 10:40 AM	MF File

File name: fourl

Files of type: All Files (*.*)

4. Algorithm Selection:



ALGORITHM SELECTION

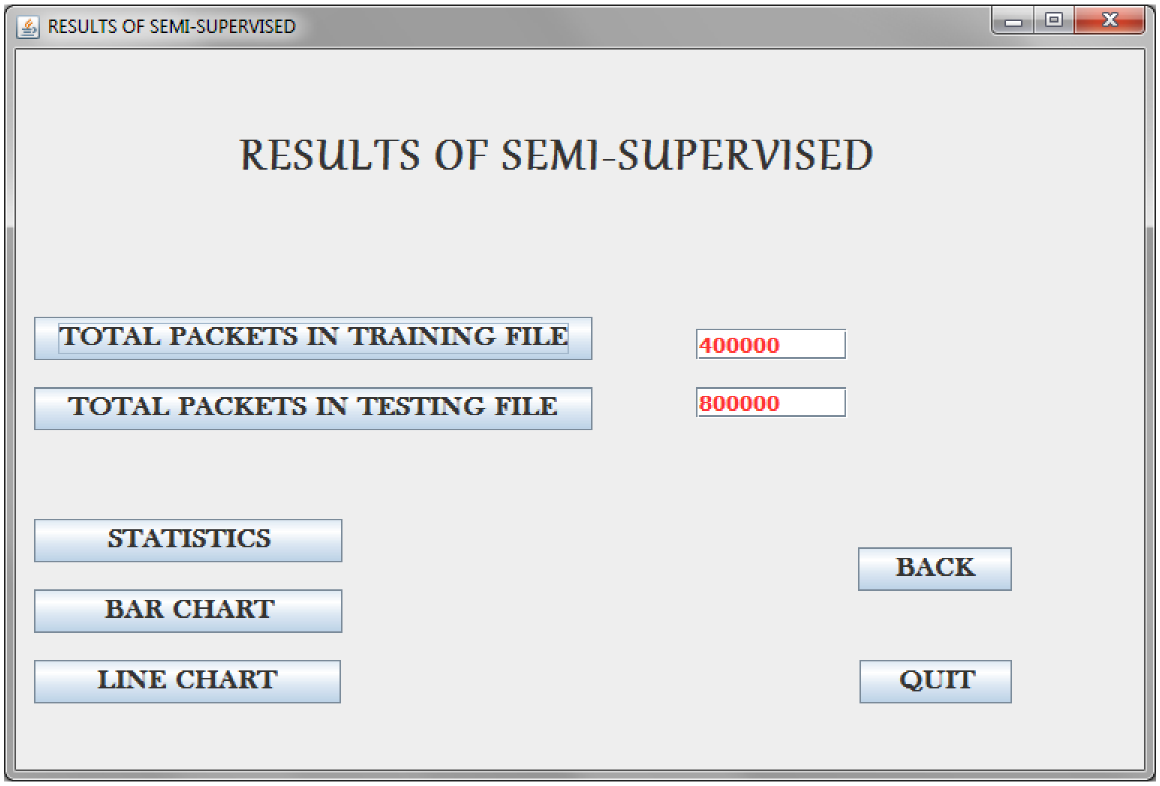
[✓ J48 ALGORITHM]

NAIVE BAYES ALGORITHM

CHOOSE DEVIATION 1.5

NO. OF ITERATIONS 3

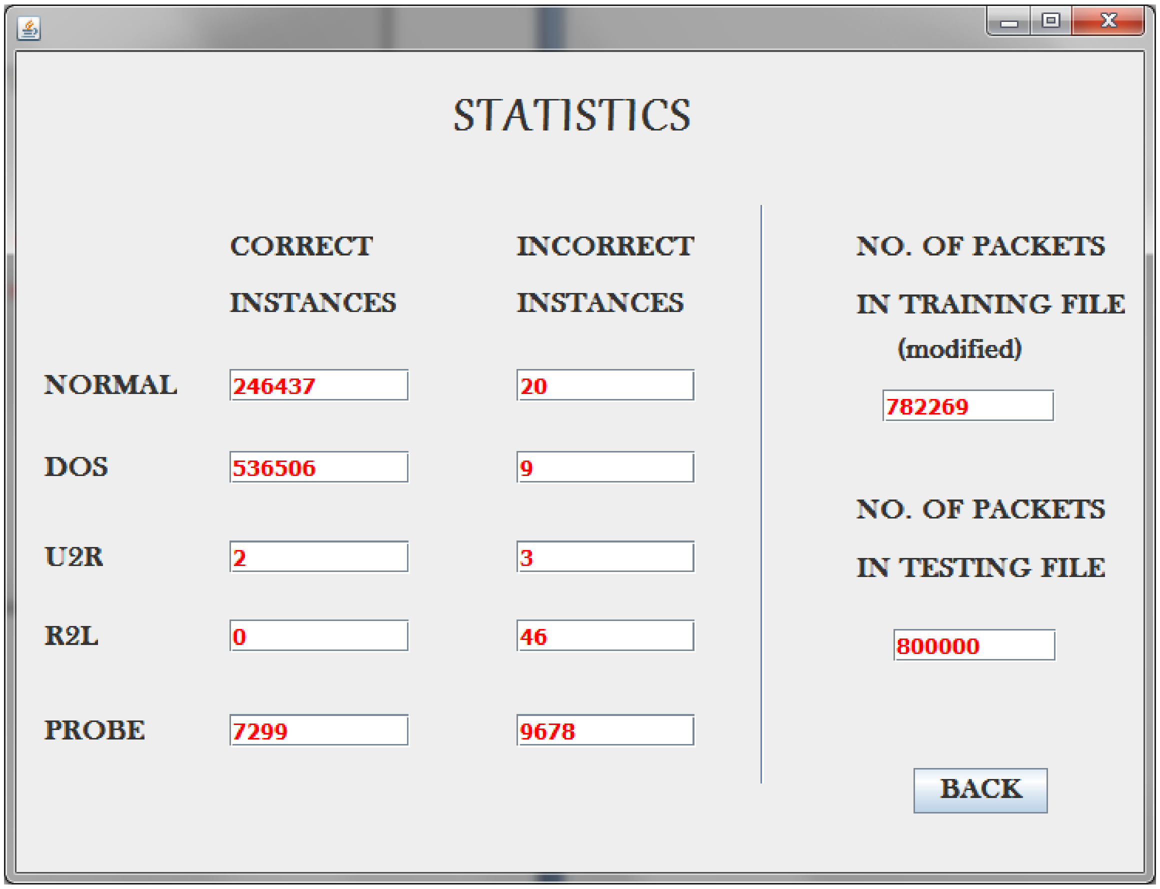
5. Result:



TOTAL PACKETS IN TRAINING FILE 400000

TOTAL PACKETS IN TESTING FILE 800000

6. Result Display in Tabular Format:



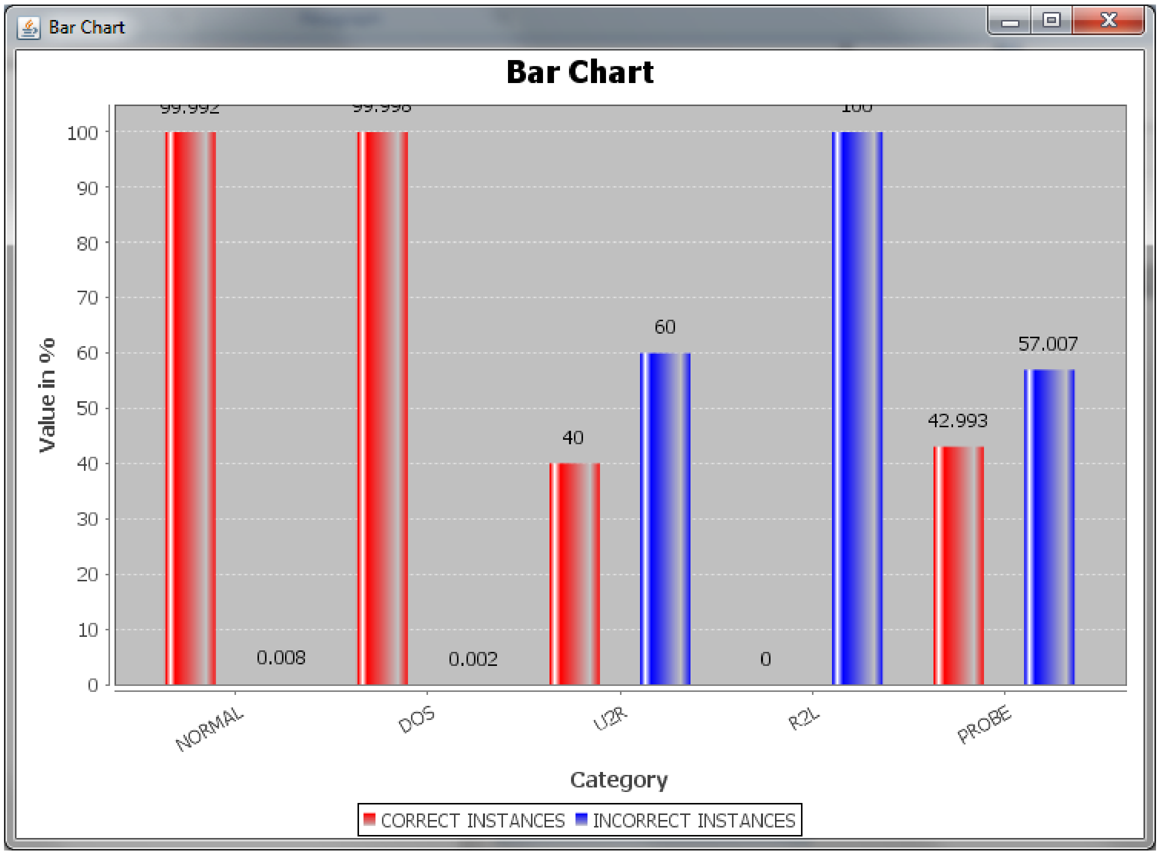
STATISTICS

	CORRECT INSTANCES	INCORRECT INSTANCES
NORMAL	246437	20
DOS	536506	9
U2R	2	3
R2L	0	46
PROBE	7299	9678

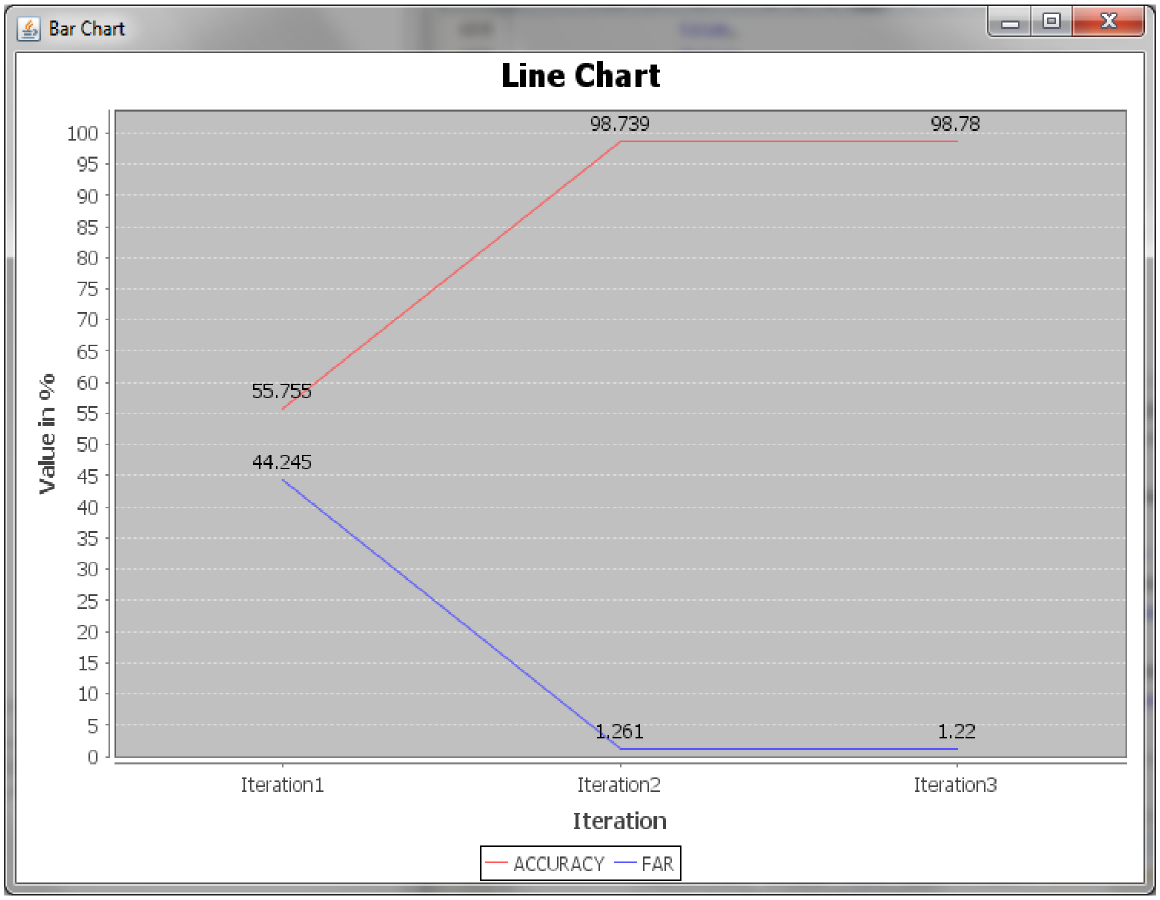
NO. OF PACKETS IN TRAINING FILE (modified): 782269

NO. OF PACKETS IN TESTING FILE: 800000

7. Result Display in Bar Graph Format:



8. Result Display in Line Chart Format:



CHAPTER 11

DEPLOYMENT AND MAINTENANCE

11.1 User Manual

11.1.1 Getting Started

The welcome shows the Project title with option 'NEXT' and 'QUIT'.

11.1.2 Browse File

Here the training data and testing data is selected by clicking the buttons 'BROWSE TRAINING FILE' and 'BROWSE TESTING FILE' respectively.

11.1.3 Browse Training File

Here the training data is selected (eg. Kddcupdata10.txt) and then the open button is clicked.

11.1.4 Algorithm

When you click on the 'Next' button after the selection of training and testing files, the page with different training algorithm appears. This page contains two option buttons namely 'J48 Algorithm' and 'Naïve Bayes'. The option for Naïve Bayes is disabled if J48 Algorithm option is chosen. At right side of the page user has to choose the deviation and the number of iterations. This page also contains the two buttons that are semi-supervised mode that performs the semi-supervised approach on the selected input files. The result button moves user to next page.

11.1.5 Result

This page displays the total number of packets in the training file and testing file. The results can also be obtained in tabular view, bar graph and pie chart format by clicking 'STATISTICS', 'BAR GRAPH' and 'PIE CHART' button respectively.

11.1.6 Exit

User can run different algorithms and compare their results.

CHAPTER 12

CONCLUSION AND FUTURE SCOPE

12.1 Conclusions

In our project, we studied different machine learning algorithms used to model intrusion detection systems, to improve detection rate and reduce False Alarm Rate. With the help of Decision Tree we built J48 algorithm and studied its performance. We used KDD Cup99 as our data set that is being used by most of researchers in the development of IDS. The performance is classified in terms of accuracy, detection rate, false alarm rate and accuracy for four categories of attack under different percentage of normal data. We are successfully reduced False Alarm Rate and increase the detection rate.

12.2 Future Scope

Future work for our project:

- Use our system for real time application in networking.
- Use the system for different datasets format.
- Use the system for detecting new attacks.
- Use more Machine Learning Algorithms to improve accuracy.

REFERENCES

[1] R. Heady, G. Luger, A. Maccabe, M. Servilla, "The Architecture of a Network Level Intrusion Detection System", *Technical report, Department of Computer Science, University of New Mexico, 1990.*

[2] A. Blum, T. Mitchell, "Combining labeled and unlabeled data with co-training", *COLT: Workshop on Computational Learning Theory, 1998.*

[3] N. D. Lawrence, M. I. Jordan, "Semi-supervised learning via Gaussian processes", *L. K. Saul, Y. Weiss and L. Bottou (Eds.), Advances in neural information processing systems 17. Cambridge, MA: MIT Press, 2005.*

[4] Xiaojin Zhu, "Semi-Supervised Learning Literature Survey", *Computer Sciences Technical Report 1530, University of Wisconsin – Madison.*

[5] C. Kemp, T. Griffiths, S. Stromsten, J. Tenenbaum, "Semi-supervised learning with trees" *Advances in Neural Information Processing System, 2003.*

[6] Yi Chien Chiu, Yuh-Jye Lee, Chien-Chung, Chang, Wen-Yang Luo, Hsiu-Chuan Huang, "Semi-supervised Learning for False Alarm Reduction", *P. Perner (Ed.): ICDM 2010, LNAI 6171, Springer-Verlag Berlin Heidelberg 2010, pp. 595–605.*

[7] Gao Xiang, Wang Min, "Applying Semi-supervised cluster algorithm for anomaly detection", *Third International Symposium on Information Processing, 978-0-7695-4261-4/10, IEEE, 2010.*

[8] Ching-Hao Mao, Hahn-Ming Lee, Devi Parikh, Tsuhan Chen, Si-Yu Huang, "Semi-Supervised Co-training and Active Learning based Approach for Multi-view Intrusion Detection", *24th Annual ACM Symposium on Applied Computing, Honolulu, Hawaii, pp. 2042-2048, 2009.*

[9] Hadi Sarvari, and Mohammad Mehdi Keikha "Improving the Accuracy of Intrusion Detection Systems by Using the combination of Machine Learning Approaches", *Published in: Soft Computing and Pattern Recognition (SoCPaR), 2010 International Conference of, Date of Conference:7-10 Dec. 2010, ISBN:978-1-4244-7897-2 ,INSPEC Accession Number:11747980.*

[10] Kamarularifin Abd Jalil, and Mohamad Noorman Masrek, "Comparison of Machine Learning Algorithm Performance in Detecting Network Intrusion", *Published in: Networking and Information Technology (ICNIT), 2010 International Conference on, Date of Conference: 11-12 June 2010, Print ISBN: 978-1-4244-7579-7, INSPEC Accession Number:11432144.*

[11] Mrutyunjaya Panda, and Manas Ranjan Patra, "Evaluating machine learning algorithms for detecting network intrusions", *International Journal of Recent Trends in Engineering 04/2009.*

[12] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani , "A Detailed Analysis of the KDD CUP 99 Data Set", *Conference: IEEE Symposium on Computational Intelligence in Security and Defense Applications - CISDA , 2009, DOI: 10.1109/CISDA.2009.5356528.*

[13] Phurivit Sangkatsanee, Naruemon Wattanapongsakorn, Chalermpol Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches", *Computer Communications 01/2011; 34:2227-2235. DOI: 10.1016/j.comcom.2011.07.00.*

[14] Sandip Sonawane, Shailendra Pardeshi and Ganesh Prasad, "A survey on intrusion detection techniques", *March 2012, World Journal of Science & Technology; 2012, Vol. 2 Issue 3, p127.*

[15] G.V. Nadiammai, S.Krishnaveni, M. Hemalatha, "A Comprehensive Analysis and study in Intrusion Detection System using Data Mining Techniques", *December 2011, International Journal of Computer Applications; Dec2011, Vol. 35, p5.*

[16] Charles Elkan, "Results of the KDD'99 Classifier Learning", *Published in: ACM SIGKDD Explorations Newsletter, Volume 1 Issue 2, January 2000.*

[17] Pachghare V.K., Kulkarni P., "Pattern Based Network security using Decision Trees and Support Vector Machine", Published in: Electronics Computer Technology (ICECT), 2011 3rd International Conference on (Volume:5), Date of Conference: 8-10 April 2011, Print ISBN: 978-1-4244-8678-6 ,INSPEC Accession Number: 12096743

APPENDIX A

Abbreviations:

IDS: Intrusion Detection System

KDD: Knowledge Discovery and Data-Mining Database

DOS: Denial of Service

U2R: User to Root

R2L: Remote to Local

RMMM: Risk Mitigation, Monitoring and Management

API: Application Programming Interface

SDLC: Software Development Life Cycle

JVM: Java Virtual Machine

JDK: Java Development Kit

JRE: Java Runtime Environment

UML: Unified Modeling Language

ARFF: Attribute-Relation File Format

APPENDIX B

Assignment 1

Project workstation selection, installations along with setup and installation report preparations

We selected Windows 7 as our workstation to build our project.
The tools we used to build our project are as follows:

- Eclipse
- Netbeans IDE

We used Java as the programming language. It requires the following setup to be installed: JDK 1.7
All the mentioned tools are open source and freely available on the internet. They are easy to install and do not take much time.
We also used Weka, a data mining tool, which is also freely available and can be downloaded from Waikato.ac.nz. We used this tool for comparing the results.

Assignment 2

Review of design and necessary corrective actions taking into consideration the feedback report of Term 1 assessment and other competitions/conferences participated like IIT, Central Universities or equivalent centers of excellence etc.

We presented a paper titled “Pattern Based Security Using Machine Learning Techniques” at the NCSEEE’14 conference at the Vishwakarma Institute of Information Technology, Pune on the 21st of February 2014. Our paper was selected and published in the Journal of Harmonized Research Issue January-March 2014.