

Chapter 1

Introduction

1.1 Overview

The importance of securing information in an organization has been very immense lately. Any organization would not want any of its information to be known by other organizations especially its competitor. In order to protect their information, organizations are willing to spent millions of dollars, showing how important the issue is. One of the solutions to securing the information is by using the Intrusion Detection System (IDS). Simply put, IDS which 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 Detection is used. The Anomaly Detection can be implemented using different other techniques such as Statistical Model, Computer Immunological Approach and Machine Learning. In this project we are using Machine Learning.

1.2 Problem Definition

The title of the project is “TO REDUCE FALSE ALARM RATE (FAR) FOR INTRUSION DETECTION SYSTEM (IDS) BY MACHINE LEARNING ALGORITHM”.

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 Applying the Software Engineering approach

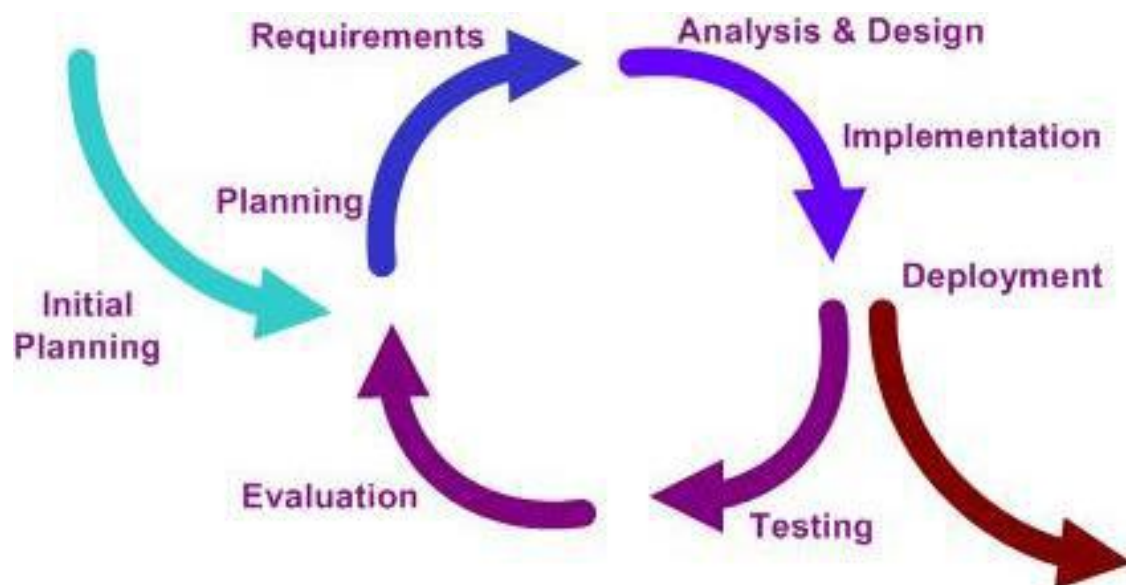


Figure 1.1 Incremental Approach

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

IEEE is a well-known organization for research work publication in Electronics and Computation field. Before starting designing phase of our project we do literature survey from various IEEE research papers and some articles from various journals. Some of them are mentioned below:

1. “Improving the Accuracy of Intrusion Detection Systems by Using the combination of Machine Learning Approaches”, Hadi Sarvari, and Mohammad Mehdi Keikha.

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.

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 a 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.

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 Tavallaei, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani,

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.

They are given some inherent problems of KDD' 99 data set that is first, for the sake of privacy, the experiments chose to synthesize both the background and the

attack data, and the data is claimed to be similar to that observed during several month of sampling data from a number of Air Force bases. However, neither analytical nor experimental validations of the data's false alarm characteristics were undertaken. Furthermore, the workload of the synthesized data does not seem to be similar to the traffic in real networks. Second, traffic collectors such as TCPdump, which is used in DARPA'98, are very likely to become overloaded and drop packets in heavy traffic load. However, there was no examination to check the possibility of the dropped packets. Third, there is no exact definition of the attacks.

They are given some deficiencies after statistical observations like redundant records and level of difficulty. For this they provided their own solution as they make their own dataset by removing all redundant data from the dataset and at last they given the conclusion that SVM is having better performance in KDDTest+.

5. “Network Intrusion Detection Using Naive Bayes”, Mrutyunjaya Panda and Manas Ranjan Patra,

In this paper, they are focused on mainly network security by Network Intrusion Detection System (NIDS), which detect attacks by observing various network activities. This paper presents the scope and the status of their research on anomaly detection. They present results on KDD CUP 99 dataset. They have proposed a framework of NIDS based on Naïve Bayes algorithm. The framework builds the patterns of the network services over data sets labeled by the services. With the built patterns, the framework detects attacks in the datasets using the Naïve Bayes Classifier algorithm.

They are given the information about the Intrusion Detection System (IDS) as Intrusion Detection System (IDS) inspects the activities in a system for suspicious behavior or patterns that may indicate system attack or misuse. Anomaly and Misuse detection are the two different categories in intrusion detection techniques. Misuse Detection is based on signatures for known attacks, so it is only as good as the database of attack signatures that it uses for comparison. Misuse detection has low false positive rate, but cannot detect novel attacks. However, anomaly detection can

detect unknown attacks, but has high false positive rate. Networking attacks are categorized into four categories DOS, Probe, U2R, R2L.

The Naïve Bayes method is based on the work of Thomas Bayes (1702-1761). In Bayesian classification, they have a hypothesis that the given data belongs to a particular class. They then calculate the probability for the hypothesis to be true. This is among the most practical approaches for certain types of problems. The approach requires only one scan of the whole data. Also, if at some stage there are additional training data, then each training example can incrementally increase/decrease the probability that a hypothesis is correct. Thus, a Bayesian network is used to model a domain containing uncertainty.

The naïve Bayes model is a heavily simplified Bayesian probability model. In this model, consider the probability of an end result given several related evidence variables. The probability of end result is encoded in the model along with the probability of the evidence variables occurring given that the end result occurs. The probability of an evidence variable given that the end result occurs is assumed to be independent of the probability of other evidence variables given that end results occur. The naïve Bayes classifier operates on a strong independence assumption. This means that the probability of one attribute does not affect the probability of the other. Given a series of n attributes, the naïve Bayes classifier makes 2^n independent assumptions. Nevertheless, the results of the naïve Bayes classifier are often correct.

They state that the error is a result of three factors: training data noise, bias, and variance. Training data noise can only be minimized by choosing good training data. The training data must be divided into various groups by the machine learning algorithm. Bias is the error due to groupings in the training data being very large. Variance is the error due to those groupings being too small.

On their experiment they are given evaluation. They carried out the experiment on 10% KDDCUP 99 dataset. They evaluate the performance of their system by the detection rate and the false positive rate. The detection rate is the number of attacks detected by the system divided by the number of attacks in the dataset. The false positive rate is the number of normal connections that are misclassified as attacks divided by the number of normal connections in the data set.

Next, they calculate the error rate, which is an estimate of the true error rate and is expected to be a good estimate, if the number of test data is large and representative of the population. It is defined as follows:

Error Rate = (Total test data — total correctly classified data) / Total test data. A “Confusion Matrix” is sometimes used to represent the result of testing. The Advantage of using this matrix is that it not only tells us how many got misclassified but also what misclassifications occurred.

Compared to the Neural Network based approach, their approach achieve higher detection rate, less time consuming and has low cost factor. However, it generates somewhat more false positives. As a naïve Bayesian network is a restricted network that has only two layers and assumes complete independence between the information nodes. This poses a limitation to that research work. In order to alleviate this problem so as to reduce the false positives, active platform or event based classification may be thought of using Bayesian network.

6. “AdaBoost Based Algorithm for Network Intrusion Detection”, Weiming Hu, Senior Member, IEEE, Wei Hu, and Steve Maybank, Senior Member, IEEE, using Artificial Intelligence in Intrusion Detection Systems”, Matti Manninen Helsinki University of Technology,

In this paper they are focused on network intrusion detection system by use of AdaBoost algorithm. They had aim to construct an intrusion detection approach with a low computational complexity, a high detection rate, and a low false-alarm rate, in this correspondence, they apply the AdaBoost algorithm to intrusion detection.

The AdaBoost algorithm is one of the most popular machine learning algorithms. Its theoretical basis is sound, and its implementation is simple. It has been applied to many pattern recognition problems, such as face recognition. However, the application of the AdaBoost algorithm to intrusion detection has not been explored so far. The AdaBoost algorithm corrects the misclassifications made by weak classifiers, and it is less susceptible to over fitting than most learning algorithms. Recognition performances of the AdaBoost based classifiers are generally encouraging. Data sets for intrusion detection are a heterogeneous mixture of categorical and continuous

types. The different feature types in such data sets make it difficult to find relations between these features. By combining the weak classifiers for continuous features and the weak classifiers for categorical features into a strong classifier, the relations between these two different types of features are handled naturally, without any forced conversions between continuous and categorical features. If simple weak classifiers are used, the AdaBoost algorithm is very fast.

The framework of their approach consists of the following four modules: feature extraction, data labeling, design of the weak classifiers, and construction of the strong classifier. For each network connection, the following three major groups of features for detecting intrusions are extracted: basic features of individual Transmission Control Protocol (TCP) connections, content features within a connection suggested by domain knowledge, and traffic features computed using a 2-s time window. Because the AdaBoost algorithm uses supervised learning, a set of data has to be labeled for training. This labeled data set should contain both normal samples labeled as “+1” and attack samples labeled as “-1.” The AdaBoost algorithm requires a group of weak classifiers designed beforehand. An individual weak classifier is simple and easy to implement. Its classification accuracy is relatively low. A strong classifier is obtained by combining the weak classifiers.

In the algorithm, decision stumps are used as weak classifiers. The decision rules are provided for both categorical and continuous features. The relations between categorical and continuous features are handled naturally, without any forced conversions between these two types of features. A simple over fitting handling is used to improve the

learning results. In the specific case of network intrusion detection, we use adaptable initial weights to make the tradeoff between the detection and false-alarm rates. There are some limitations in their algorithm. It is noted that the implementation of their AdaBoost-based intrusion detection algorithm is not amenable to incremental learning. When the system changes over time, the newly produced data should be labeled and merged with the previous sample data, and the classifier is retrained using the merged sample data. Although our algorithm has a very low computational complexity, which makes it possible to frequently retrain the classifier, offline

learning is still a limitation when it is necessary to adapt to complicated and changing network environments.

The experiment results show that their algorithm has a very low false-alarm rate with a high detection rate, and the run speed of our algorithm is faster in the learning stage compared with the published run speeds of the existing algorithms. The experimental results illustrate that their algorithm has a competitive performance, as compared with the published intrusion detection algorithms, as tested on the benchmark sample data.

7. “Results of the KDD’99 Classifier Learning”, Charles Elkan,

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."

8. "Naive Bayes Classification of Uncertain Data", Jiangtao Ren, Sau Dan Lee, Xianlu Chen, Ben Kao, Reynold Cheng and David Cheung,

In this paper they are focused on classification of uncertain data rather than precise data by Naïve Bayes classification. In many emerging applications, however, the data is inherently uncertain. Sampling errors and instrument errors are both sources of uncertainty, and data are typically represented by probability distributions rather than by deterministic values. Data uncertainty arises naturally in many applications due to various reasons. In this paper we study the problem of classifying objects with multi-dimensional uncertainty. In particular, an object is not a simple point in space, but is represented by an uncertainty region over which a pdf is defined. Formally, they consider a set of n objects in a d -dimensional space. The location of each object is represented by a pdf p that specifies the probability density of each possible location. They assume that the pdf of each tuple is independent of the others.

Naive Bayes is a widely used classification method based on Bayes theory. Based on class conditional density estimation and class prior probability, the posterior class probability of a test data point can be derived and the test data will be assigned to the class with the maximum posterior class probability. The key problem in naive Bayes method is the class conditional density estimation. Traditionally the class conditional density is estimated based on data points. For uncertain classification problems, however, we should learn the class conditional density from uncertain data objects represented by probability distributions. In order to extend the naïve Bayes method to handle uncertain data, they proposed three methods in this paper: Averaging (AVG), Sample-based method (SBC) and Formula-based method (FBC).

They introduced some related works about uncertain data mining, uncertain data classification, naïve Bayes model and kernel density estimation. They proposed two approaches for handling uncertain data in naive Bayes classification problem. One is averaging and the other is distribution-based. The key problem in naive Bayes model is class conditional probability estimation, and kernel density estimation is a

common way for that. They had extended the kernel density estimation method to handle uncertain data. This reduces the problem to the evaluation of double-integrals. For particular kernel functions and probability distributions, the double integral can be analytically evaluated to give a closed-form formula, allowing an efficient formula-based algorithm. In general, however, the double integral cannot be simplified in closed forms. In this case, a sample-based approach is proposed. In sample based approach, every training and testing uncertain data object is represented by sample points based on their own distributions. When using kernel density estimation for a data object, every sample point contributes to the density estimation. The integral of density can be transformed into the summation of the data points' contribution with their probability as weights.

On their experiments they are given conclusion that extensive experiments on several UCI datasets show that the uncertain naive Bayes model considering the full pdf information of uncertain data can produce classifiers with higher accuracy than the traditional model using the mean as the representative value of uncertain data. Time complexity analysis and performance analysis based on experiments show that the formula-based approach has great advantages over the sample-based approach.

9. “Naive Bayesian Classification of Structured Data”, Peter A. Flach, Nicolas Lachiche.

In this paper, they are focused on Naïve Bayes classification of structured data. They present 1BC and 1BC2, two systems that perform naive Bayesian classification of

structured individuals. This paper provides an unified view of the 1BC and 1BC2 approaches. In particular, each approach has been presented as one way of upgrading the propositional naive Bayesian classifier. Compared to others, a context dependency has been added that takes into account explicitly the structure of the object, and enables 1BC2 to deal with objects involving similar components at different locations in their structure. That context is the key to a recursive implementation of 1BC, compared to the previous non-recursive implementation. Now 1BC inherits the efficiency of 1BC2, which is much better than the typical propositionalisation

approach. Propositionalisation (i.e., mapping to a different feature space) is now performed only implicitly, in the spirit of kernel methods.

They are presented two approaches to upgrade the propositional naive Bayesian classifier to deal with structured data. One approach is based on upgrading attributes to first-order features, i.e., it works in the language space. This approach has been implemented in the 1BC system. The second approach generalized the naive Bayes estimate of the probability of a tuple of elements from probabilities of its components, to collections of elements. It works in the individual space, and has led to the 1BC2 system. Both systems are available on-line. They have compared the two systems both with respect to the probabilities they compute, and regarding their performance on artificially constructed data sets. These experiments confirm that 1BC2 outperforms 1BC on problems where modeling the cardinality of collections or the distributions of elements of collections is important. On the other hand, they also demonstrate that 1BC2 requires more data to estimate these probabilities reliably. Experimental results have been obtained on a range of benchmark ILP data sets. They confirm that the two systems perform comparably to other ILP systems.

Probabilistic classifiers are often criticized because they do not induce declarative, interpretable models. However, an often overlooked advantage is that the probabilistic model is a ranker rather than a classifier, and thus can be calibrated to improve classification performance. Using ROC analysis we have shown that the default decision threshold used to turn the ranker into a classifier is often sub-optimal. They suggest an algorithm to improve the accuracy and cost of a probabilistic classifier on two-class and multi-class problems. This includes both the 1BC and 1BC2 systems and works on structured data as well as on attribute-value data. In the propositional case, the naive Bayesian decomposition leads to loss of probabilistic information but not of logical information. In the relational case, logical information is lost as well. This means that relational naive Bayesian classifiers such as the ones proposed in this paper can be expected to be sensitive to the representation. In our experiments so far we have not paid particular attention to this, and have worked with the simplest or most obvious representation.

10. “A Bayesian Networks in Intrusion Detection Systems”, M. Mehdi, S. Zair, A. Anou and M. Bensebti.

In this paper, they are focused on network intrusion detection system. Intrusion detection systems (IDS) have been widely used to overcome security threats in computer networks. Anomaly-based approaches have the advantage of being able to detect previously unknown attacks, but they suffer from the difficulty of building robust models of acceptable behavior which may result in a large number of false alarms caused by incorrect classification of events in current systems.

They proposed a new approach of an anomaly Intrusion detection system (IDS). It consists of building a reference behavior model and the use of a Bayesian classification procedure associated to unsupervised learning algorithm to evaluate the deviation between current and reference behavior. Continuous re-estimation of model parameters allows for real time operation. The use of recursive Log-likelihood and entropy estimation as a measure for monitoring model degradation related with behavior changes and the associated model update show that the accuracy of the event classification process is significantly improved using our proposed approach for reducing the missing alarm.

This probabilistic approach is a tentative research on the possibility of applying Bayesian unsupervised learning in the detection of network intrusions. Based on unsupervised learning algorithms (Bayesian classification), some novel detection methods are proposed showing a very high detection rate with a reasonable true positive rate as results, much better than the classic method. By training with unsupervised learning algorithms, namely, Bayesian classification procedure, the log analyzer performs well in discovering the inherent nature of the dataset, clustering similar instances into the same classes.

They had presented a new anomaly IDS design using a parametric mixture model for behavior modeling and Bayesian based detection. Continuous model update is accomplished by model parameter re-estimation. Algorithms for detection and update phases are designed for real-time operations. Preliminary experimentations show that proposed algorithms have some limitations such as that the kernel distributions are used to model numerical data with continuous and unbounded nature, the Gaussian parametrical model may not be suitable for complex data and that the

use of mixed models assumes statistical independence between trials, which can be restrictive in some cases. Despite these drawbacks the system presents real-time feasibility with no special hardware requirement. Moreover, it is being extended to detect security violations in a heterogeneous networked environment. The scalability, performance and fault tolerance can be improved when mobile agents perform distributed detection and do not need a central location where data is gathered. For instance, for wireless networks such as Mobile Ad hoc Networks are greatly prone to security threats. Since intrusion to the transmission support is relatively easy, compared to fixed networks, the use of such IDS is greatly recommended.

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 ages of this detection is that it has high

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 noval 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 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 which 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 which 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 with in 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.

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 which 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

	Four categories of features
TCP connectionbasic Characteristic(1~9)	duration, protocol_type, service, flag, src_bytes,dst_bytes, land,wrong_fragment, urgent
TCP connectioncontent Characteristic(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
Time-basednetwork traffic(23~31)	count, srv_count, serror_rate, srv_seror_rate,error_rate, rv_rerror_rate, same_srv_rate,diff_srv_rate, srv_diff_host_rate
Host-basednetwork 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_sr v_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 above table is mainly used in K-NN algorithm. 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 above table 3.2 is helpful in KNN algorithm. 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:

- Display result in Table view.
- Display result in Graphical view.
- Display result in Percentile view.
- There will be different graphical view options.

The program python to be used will also act as our front end in the system. The starting page will allow user to input data in .tab and .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

1. Python 2.7 Interpreter
2. Linux (UBUNTU 11.4) or Windows XP, Vista.

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 which 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

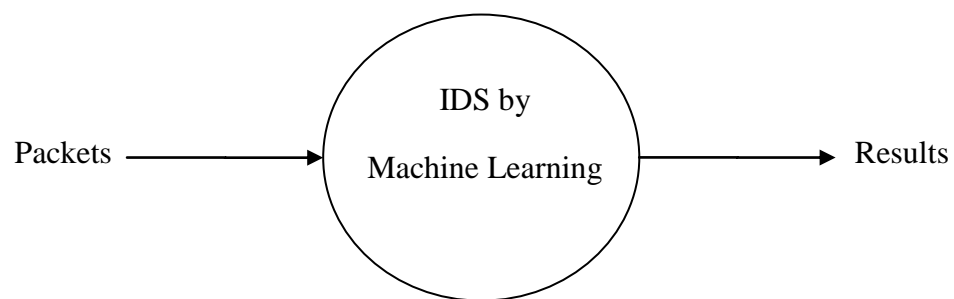


Figure 3.1 DFD Level-0

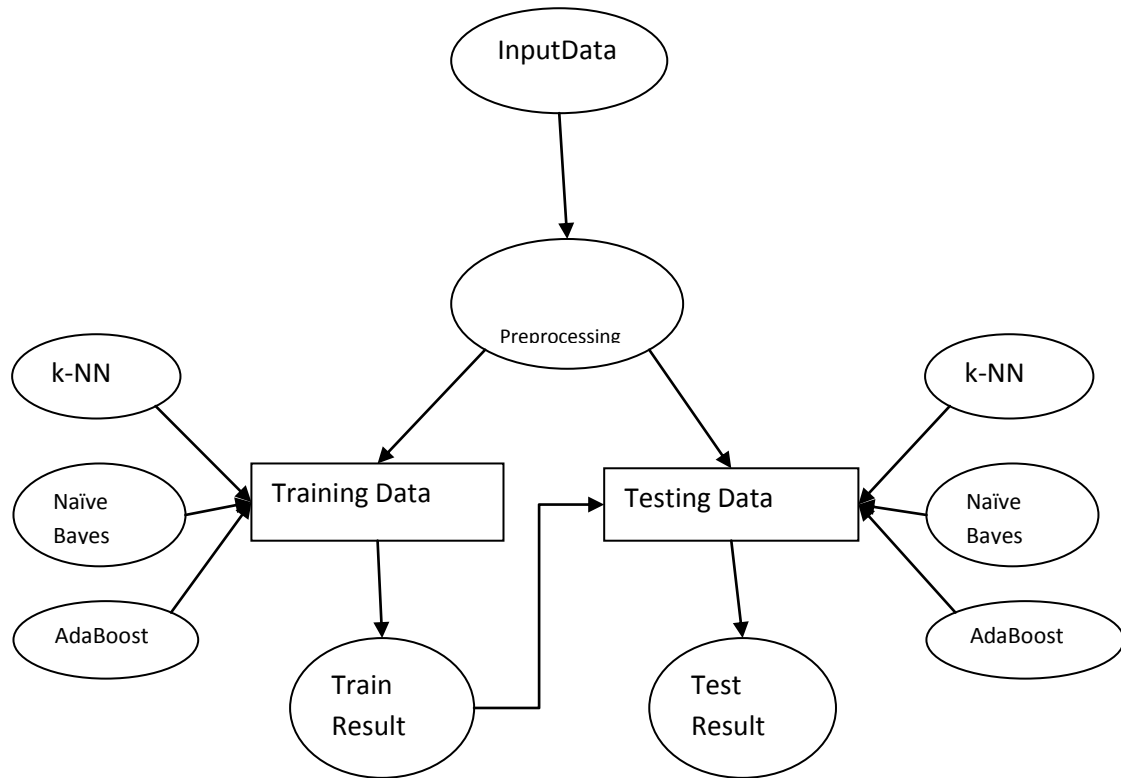


Figure 3.2 DFD Level -1

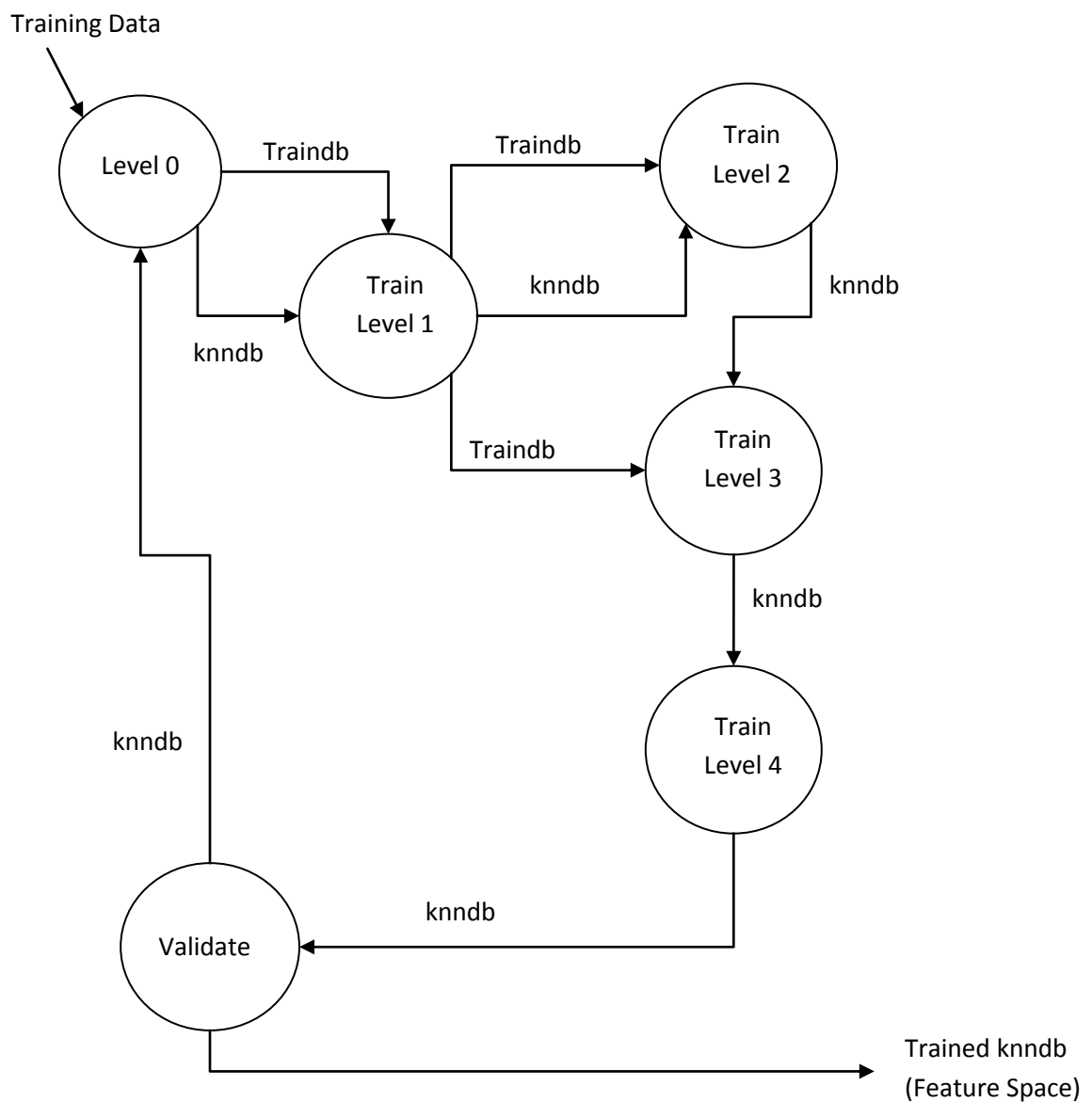


Figure 3.3 DFD Level-2 k-NN Training

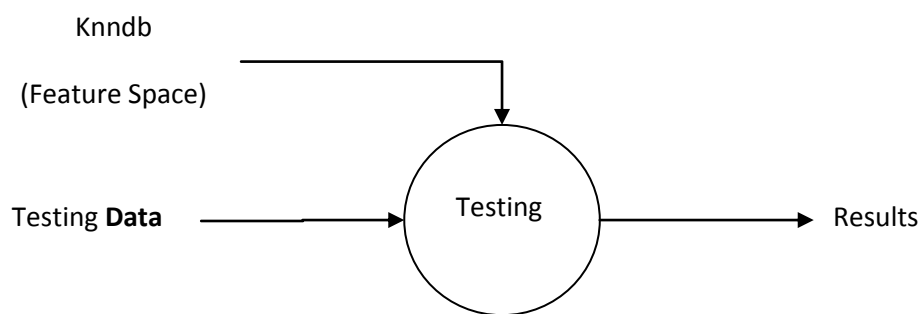


Figure 3.4 DFD Level-2 k-NN Testing

3.5.2 State Transition Diagram

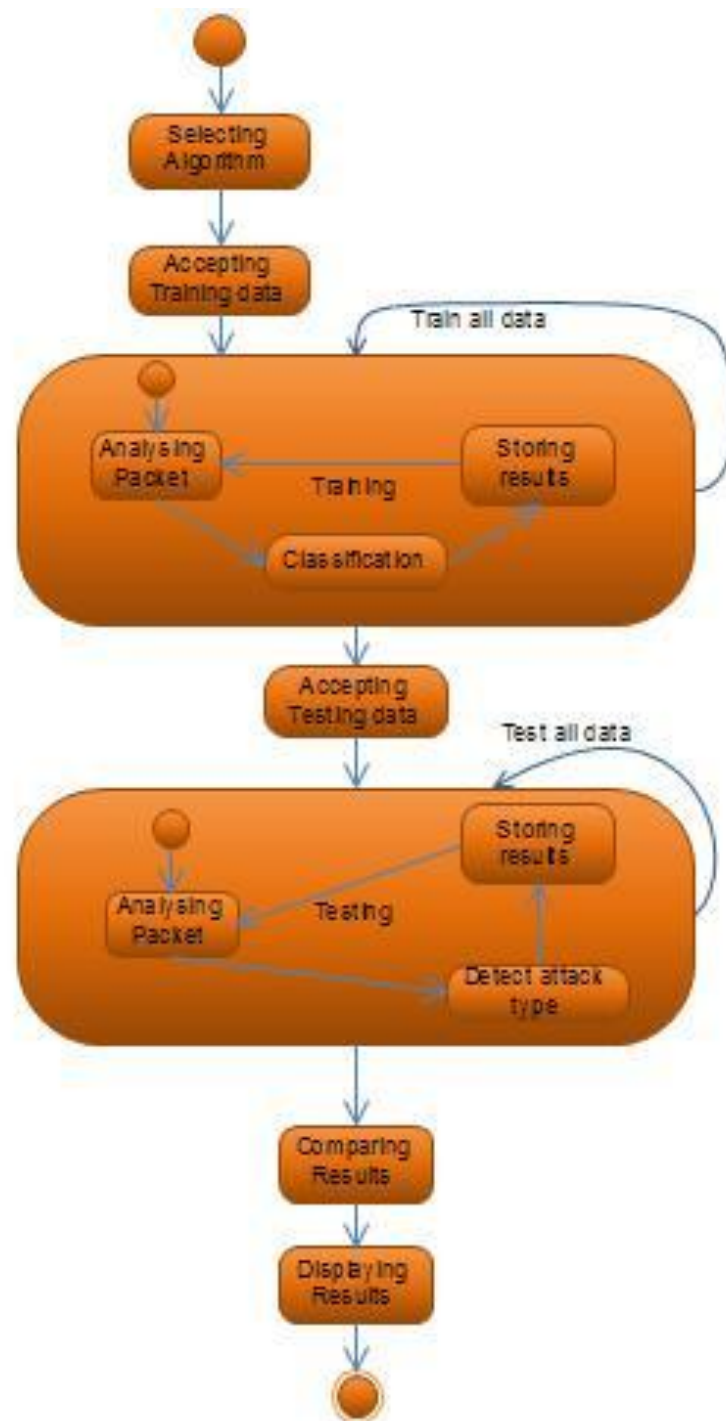


Figure 3.5 State Machine Diagram

Chapter 4

System Design

4.1 System Architecture

The intrusion detection system can analyze the dataset captured packets 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.

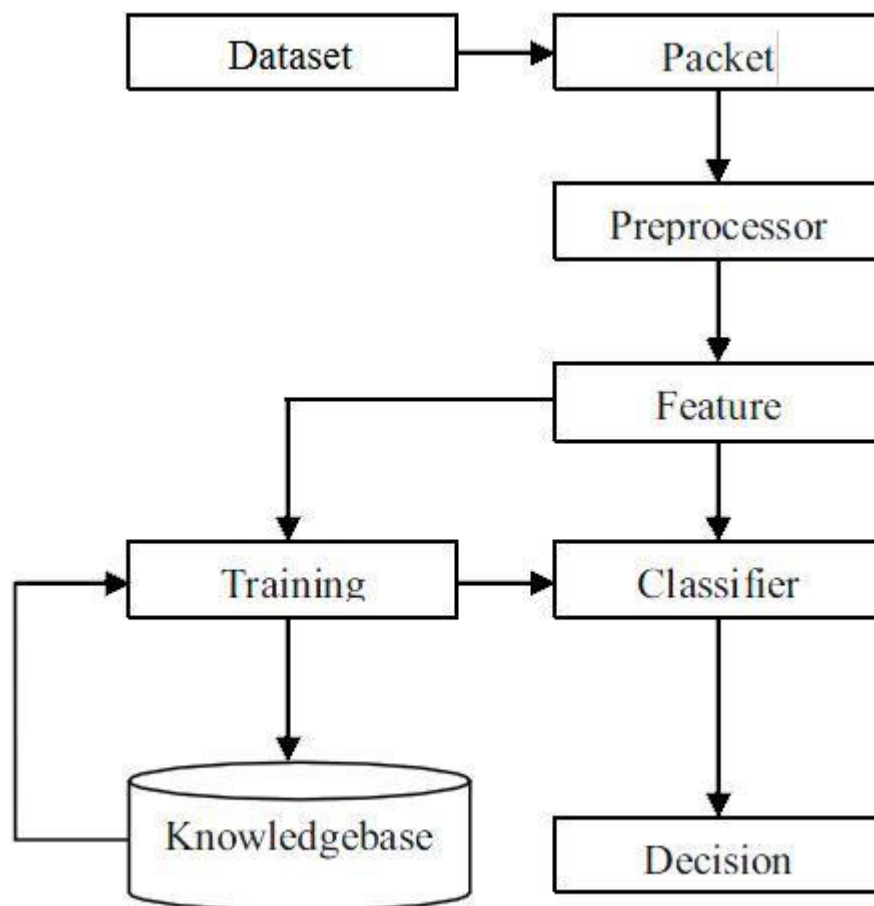


Figure1. Block diagram of system overview

Based on the figure above, there are several modules that introduce the intrusion detection system based on the artificial neural networks architecture. Now we are about to describe this layering structure step by step.

1. Packet Monitor: This module capture packets from dataset to serve for the data source of the IDS.
2. Preprocessor: In preprocessing phase, dataset collected and processed for use as input to the system.
3. Feature Extractor: This module extracts feature vector from the packets and submits the feature vector to the classifier module.
4. Classifier: The function of this module is to analyze the packet stream and to draw a conclusion whether intrusion happens or not.
5. Decision: When detecting that intrusion happens, this module will send a warning message to the user.
6. Knowledgebase: This module serves for the training samples of the classifier phase. The intrusion samples can be perfected under user participation, so the capability of the detection can improve continually.

All of these modules together make the IDS architecture system based on the Machine learning techniques, and the function of each module has been introduced briefly. The present study is aimed to solve a multi class problem in which not only the attack records are distinguished from normal ones, but also the attack type is identified.

4.2 UML Diagrams

4.2.1 Use Case Diagram

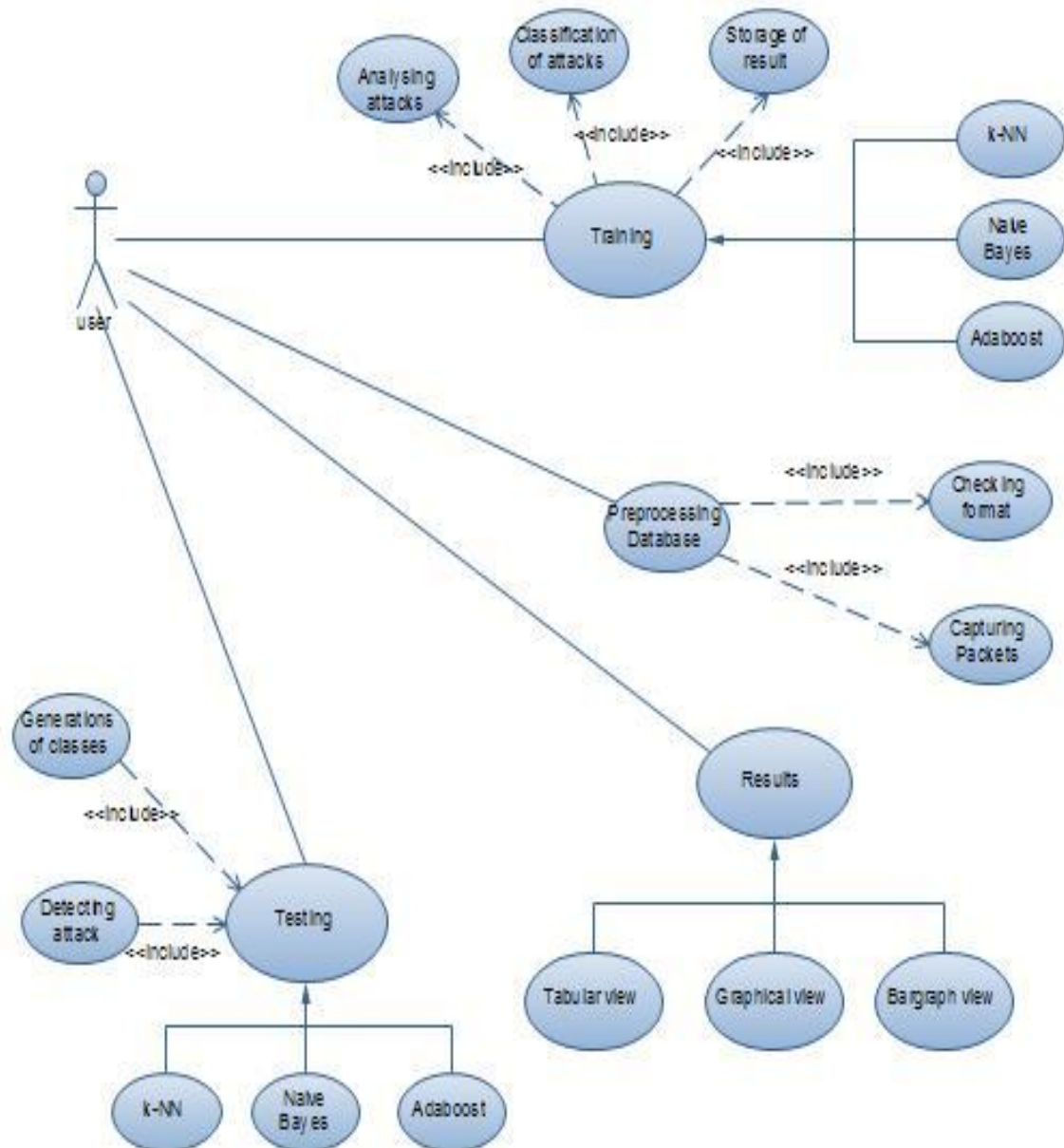


Figure 4.1 Use Case Diagram

4.2.2 Sequence Diagram

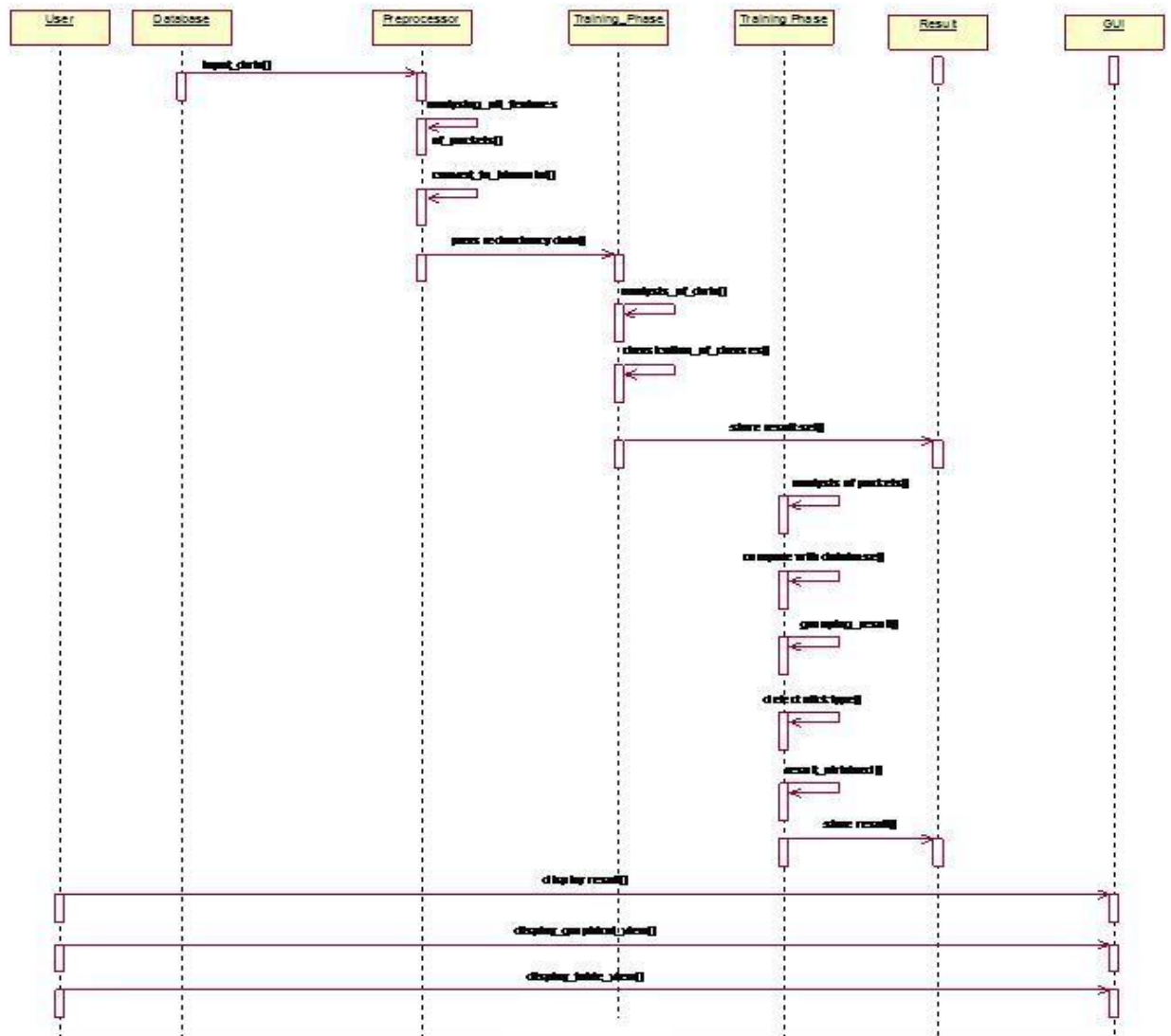


Figure 4.2 Sequence Diagram

4.2.3 Activity Diagram

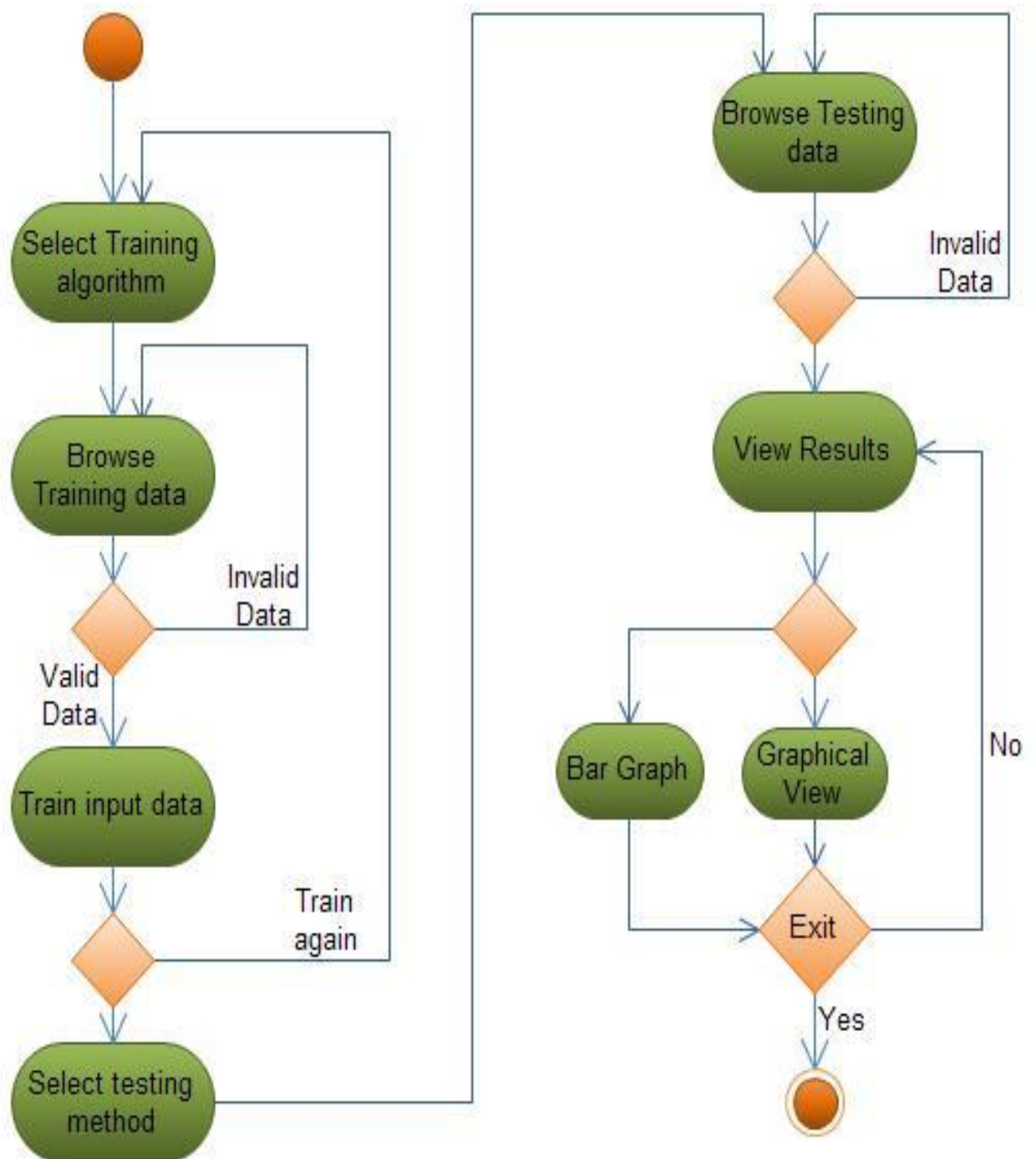


Figure 4.3 Activity Diagram

Chapter 5

Technical Specification

5.1 Domain Area of Project

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

5.2 Technology Consideration

1. Linux (Ubuntu 11.04)
2. Python 2.7
3. Python Packages -
 - NumPy
 - matplotlib
 - PyGTK
 - ReportLab
 - Imaging

5.2.1 Python 2.7

Python is a high-level, interpreted, interactive and object oriented-scripting language.

Python was designed to be highly readable which uses English keywords frequently where as other languages use punctuation and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** This means that it is processed at runtime by the interpreter and you do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** This means that you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** This means that Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is Beginner's Language:** Python is a great language for the beginner programmers and supports the development of a wide range of applications, from simple text processing to WWW browsers to games.

History of Python:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted, Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing it's progress.

Python Features:

Python's feature highlights include:

- **Easy-to-learn:** Python has relatively few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language in a relatively short period of time.
- **Easy-to-read:** Python code is much more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's success is that its source code is fairly easy-to-maintain.
- **A broad standard library:** One of Python's greatest strengths is the bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Support for an interactive mode in which you can enter results from a terminal right to the language, allowing interactive testing and debugging of snippets of code.

- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above mentioned features, Python has a big list of good features, few are listed below:

- Support for functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- Very high-level dynamic data types and supports dynamic type checking.
- Supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

5.2.2 Python Packages

1. NumPy

NumPy is the fundamental package needed for scientific computing with Python. It contains:

- a powerful N-dimensional array object
- sophisticated broadcasting functions
- basic linear algebra functions
- basic Fourier transforms
- sophisticated random number capabilities
- tools for integrating Fortran code.
- tools for integrating C/C++ code

2. Matplotlib

Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in python scripts, the python and ipython shell (ala matlab or mathematica), web application servers, and six graphical user interface toolkits. matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc, with just a few lines of code.

3. PyGTK

PyGTK lets you easily create applications with a graphical user interface using the Python programming language. PyGTK uses the GTK+ toolkit to offer a comprehensive set of graphical elements and other useful programming facilities, like the handling of internationalized text (bidirectional text or non-Latin glyph sets), first class accessibility support through the Gnome Accessibility Framework (ATK library). The current version of PyGTK requires at least GTK+ version 2.8.0.

PyGTK applications are truly multiplatform and they're able to run, unmodified, on Linux, Windows, MacOS X and other platforms. You can see some of the applications where PyGTK is being used, which range from small single-purpose scripts up to large full featured applications.

4. ReportLab

ReportLab gives Python programs the power to output directly in Adobe's PDF format. The open source version is fully functional in the hands of a Python programmer. Useful for publishing course materials.

5. Imaging

Python Imaging Library, the Python Imaging Library (PIL) adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and powerful image processing capabilities

Chapter 6

Project Estimate, Schedule and team structure

6.1 Feasibility Study

The feasibility study of our project “To Reduce False Alarm Rate [FAR] for Intrusion Detection System [IDS] by machine learning algorithm” involves the following constraints:

6.1.1 Technical Feasibility

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

1. Hardware Components
 - Computer system
2. Software Components
 - Python 2.7, Linux (Ubuntu 11.04)

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.

6.1.2 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:

Component	Cost
Personal Computer (Intel Core2Duo CPU 2.5 GHz , 1 GB RAM)	Rs. 25,000
Linux (Ubuntu 11.04) Operating System	Rs. 8,650
Python 2.7	Rs. 0

Table 6.1 Component Cost

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

6.1.3 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.

6.1.4 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.

6.2 Resources

6.2.1 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.

6.2.2 System Requirements

1. Hardware Requirements

“To Reduce False Alarm Rate [FAR] for Intrusion Detection System [IDS] by machine learning algorithm” for Linux has following minimum hardware requirements.

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

2. Software Requirements

“To Reduce False Alarm Rate [FAR] for Intrusion Detection System [IDS] by machine learning algorithm” has the following software requirements.

- Linux (Ubuntu 11.04)
- Python 2.7

6.3 Scheduling

Project Start Date	26-July-2011
Project End Date	21-April-2012
Project Duration	9 months

Table 6.2 Project Duration

Sr. No.	Milestone Name	Milestone Description Information relating to the deliverable at this milestone	Timeline number of weeks for the current milestone	Remarks Comments such as the weight age of the milestone in percentage of the total project etc.
1	Design of Project Overview Construction	Shows the participants in the project and an overall design.	11	15%

2	Developing and testing of modules	Preparing and designing the User Interface, Command, Syntax and Semantic Recognition, Command Implementation, and additional utilities	16	50%
3	Final Integration and Project testing	Testing of project and specification	2	30%
4	Documentation and Project report and Presentation		2	5%

Table 6.3 Phases of Project

6.4 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, assess 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.

6.4.1 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.

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

Table 6.4 Risk Table

6.4.2 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.

Team Structure

Sr. No.	Work	Team members
1	GUI	Anish Kumar, Bhushan Desai
2	KNN	Gaurav Neelwarna
3	Naïve Bayes	Ashutosh Kakade
4	AdaBoost	Gaurav Neelwarna
5	Documentation	Gaurav Neelwarna, Bhushan Desai, Anish Kumar, Ashutosh Kakade

Chapter 7

Software Implementation

7.1 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 K-NN, Naïve Bayes and AdaBoost. During implementation proposed CPU time, 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 PYTHON which is OS independent and pure object oriented.

7.2 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.

7.3 Important Modules and Algorithms

7.3.1 k-NN Algorithm

7.3.1.1 Introduction

The k-nearest neighbours algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. k-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common amongst its k nearest neighbours (k is a positive integer, typically small). The main idea is to choose heuristically optimal k nearest neighbours by cross validation technique. If $k = 1$, then the object is simply assigned to the class of its nearest neighbour.

The basic example of k-NN is as shown in Figure1 which depicts a 3-Nearest Neighbour Classifier on two-class problem in a two-dimensional feature space. In this example the decision for triangle1 is straightforward – all three of its nearest neighbours are of class O so it is classified as an O. The situation for triangle0 is a bit more complicated at it has two neighbours of class star and one of class O. This can be resolved by simple majority voting or by distance weighted voting.

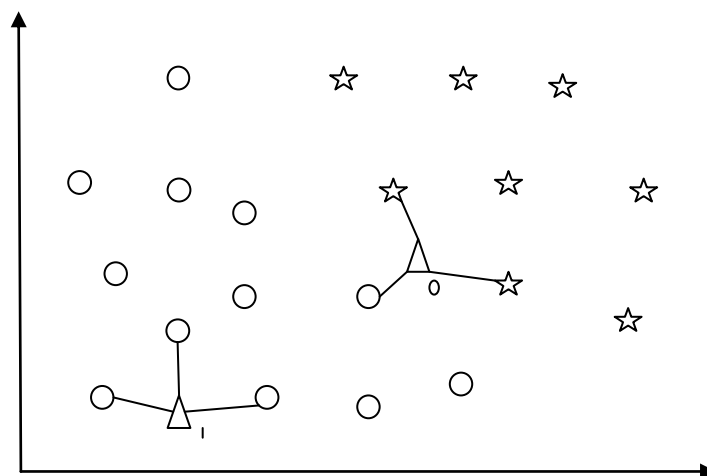


Fig 7.1 Representation of 3NN classifier on a two-class problem

7.3.1.2 Generalization:

There are total 22 types of sub-attacks given in KDD'99 Data Set, classified into 4 attack types. So, we need generalize our training data-entries. After generalizing, all data-entries will have their attack types attach to them, according to table shown below and all entries are stored back into training database.

Sr. No.	Four Attack Categories			
	Probe	DoS	U2R	R2L
#1	ipsweep	Back	buffer overflow	Ftpwrite
#2	Nmap	Land	Loadmodule	guess passwd
#3	portsweep	neptune	perl	Imap
#4	Satan	Pod	rootkit	Multihop
#5	-	Smurf	-	Phf
#6	-	teardrop	-	Spy
#7	-		-	Warezcclient
#8	-		-	Warezmater

Table 7.1 Attack Types Classification

7.3.1.3 Filter:

Filter is used to delete duplicate values from training data. It is coded into module filterdb.py. And these filtered entries are stored back into training data.

7.3.2 Training:

In training phase, user has to set following parameters:

- ⤴ Split/don't split knndb format (Feature Space)
- ⤴ k-NN Classifier
- ⤴ Weighting Function
- ⤴ Error Acceptance Value (EAV)

7.3.2.1 Don't split /Split knndb format (Feature Space)

This parameter selects the storage structure of feature space. If user selects 'Don't split knndb format' then whole feature space is stored into one single file, and 'k' nearest neighbours are being searched into this file.

If user selects 'Split knndb format' then whole feature space is divided according to its results, that is Normal, Dos, Probe, U2r, R2l, and they are stored into different files. And 'k' nearest neighbours are being searched into all files, so we get maximum 'km' neighbours where 'm' is the number of categories.

Mathematically,

$$\text{knndb_format} = \begin{cases} 1, & \text{if user selects 'Don't split knndb'} \\ 2, & \text{if user selects 'Split knndb'} \end{cases}$$

7.3.2.2 k-NN Classifier:

This parameter gets value of 'k' from user. User can select any positive integer value ranging from 1 to 9.

Mathematically,

$$k = \begin{cases} n, & \text{if } k > n \\ k, & \text{Otherwise} \end{cases}$$

Where, n is number of packets in knndb.

7.3.2.3 Weighting Function:

This parameter selects the weighting function from user. It has following weighting formulae available:

1. Absolute Distance
2. Eucliden Distance
3. 3-Norm Manhattan Distance
4. 4-Norm Manhattan Distance

Mathematically,

$$\text{Weighting Function} = \begin{cases} 1, & \text{if 'Absolute Distance'} \\ 2, & \text{if 'EuclideanDistance'} \\ 3, & \text{if '3 - Norm Manhattan Distance'} \\ 4, & \text{if '4 - Norm Manhattan Distance'} \end{cases}$$

7.3.2.4 Error Acceptance Value (EAV):

This parameter selects the error acceptance value from user. Error acceptance value gets maximum error value, which system should achieve. It accepts positive integer value from user.

After setting these parameters it asks user to select training file or enter training file path. This file should be in correct format. After entering file path, system checks following conditions in sequence:

1. Check availability of file.
2. If file is available then check its format.
3. If format is correct then checks format of every data-entry.

If any error occurred then it prints proper error message and asks user to select file again.

Perform generalization and filter on training data. Generalization changes the packet types from sub-attacks to attack and stores the data-entries back into training data. Filter delete duplicate entries from training data and store it back into training data.

After setting all parameters and generalization and filtering training algorithm is executed:

7.3.3 Training Algorithm:

Steps **Steps**
No.

Assumptions:

traindb = training data file.

'knndb' is feature space.

N = Total number of packets in training data file.

1 Initialize, cycle=1

2 [Level 0:]

 If cycle = 1 then

2.1 If knndb_format = 1 then

 If $k > N$ then

 Assign $k = N$

 Until $k < N$, where N is total number of packets in training data.

 Move first 'k' entries from training data to knndb.

 Store values of k_i into file knndb_info.txt

2.2 Else if knndb_format = 2 then

 For each packet type 'i'

 if $k_i > n_i$, where n is total number of entries of type i. then

 Assign $k_i = n_i$.

 Until $k_i < N$, where N is total number of packets in training data.

 Move first 'k' entries of type i into knndb of type i.

 Store values of k_i into file split_knndb_info.txt

 Store parameters into file input.txt.

2.3 Else

 Store knndb_level4 into knndb

3 [Level 1]

3.1 Initialize, Data Structures -

Level1_round1

Level1_round2

Level1_round3

Knndb

3.2 Read training data file into traindb and knndb file (Feature Space) into knndb

Print Number of packets need to be processed.

Print Number of packets of each type 'i'.

3.3 For each packet in traindb:

Classify 'packet' using knndb

3.3.1 If correctly classified then

Add packet to level1_round2

3.3.2 Else

Add packet to -

level1_round1

level1_round3

Knndb

3.4 Store -

Data Structure – File

Knndb - knndb_level1

level1_round1 – level1_round1

level1_round2 – level1_round2

level1_round3 – level1_round3

4 [Level 2]

4.1 Initialize, Data Structure -

```

    level2_round1
    Knndb
4.2    Read knndb_level1 into traindb and level1_round3 into knndb.
    Copy knndb into temp_knndb
4.3    For each 'packet' into temp_knndb:
        Delete 'packet' from knndb
        Classify packet using knndb
4.3.1    If correctly classified then
            Discard packet
4.3.2    Else
            Add packet to -
                level2_round1
                Knndb
4.4    Store -
        Data Structure – File
        knndb          - knndb_level2
        level2_round1 – level2_round1
4.5    Print Number of packets processed, Discarded and Added into knndb.
5      [Level 3]
5.1    Initialize, Data Structures -
        level3_round1
5.2    Read knndb_level2 into knndb and level1_round2 into traindb.
5.3    For each 'packet' in traindb
        Classify packet using knndb
5.3.1    If correctly classified then
            Discard packet
```

- 5.3.2 Else
 - Add packet to -
 - level3_round1
 - Knndb
- 5.4 Store -
 - Data Structure – File
 - knndb - knndb_level3
 - level3_round1 – level3_round1
- 5.5 Print Number of packets processed, Discarded, Added into knndb.
- 6 [Level 4]
- 6.1 Read knndb_level3 into knndb.
 - Copy knndb into temp_knndb
- 6.2 For each 'packet' into temp_knndb
 - Delete packet from knndb
 - Classify packet into knndb using 50-NN
- 6.2.1 If correctly classified then
 - Discard packet
- 6.2.2 Else
 - Add packet to knndb
- 7 Read Number of packets from file 'level2_round1' into error1 and
 Number of packets from file 'level3_round1' into error2
- 7.1 If error1 and error2 < EAV (Error Acceptance Value) then
 - Goto step 8
- 7.2 Else

```
Cycle = cycle + 1  
Goto step 2  
8      Save data structure 'knndb_level4' and contents of file 'input.txt' into files  
      having its name  
      Structure as a knn parameters.  
9      Quit
```

7.3.4 Determining Nearest Neighbors:

For determining Nearest Neighbors four weighting functions are given, as follows-

1. Absolute Distance
2. Euclidean Distance
3. 3-Norm Manhattan Distance
4. 4-Norm Manhattan

Functions of these weighting functions according to knndb_format are given into module distance.pyc and from there they are called.

7.3.5 Classification:

Packets are classified according to the k-NN classifier and weighting function selected by user. While classification, k nearest neighbours are determined based on the weighting function. For this distance of each given packet-entry is determined with each entry present in feature space, and k nearest entries are determined including their packet-types. From this result of given packet-entry is determined.

7.3.6 Modules:

For implementation of k-Nearest Neighbour algorithm we create following modules -

Sr. No.	Module Name	Description
1.	distance.py	This module contains functions for distance calculation.
2.	filterdb.py	This module contains function for filtering given data-set.
3.	generalizer.py	This module contains function for generalizing given data-set.
4.	Knn.py	This module contains functions for determining nearest neighbors.
5.	Level0.py	This module contains function to manage pre-processing on data-set.
6.	packetformat.py	This module contains function to convert data-entries into required format for further computation.
7.	Trainlevel1.py	This module contains function to execute code in step 3 of training algorithm.
8.	Trainlevel2.py	This module contains function to execute code in step 4 of training algorithm.
9.	Trainlevel3.py	This module contains function to execute code in step 5 of training algorithm.
10.	Trainlevel4.py	This module contains function to execute code in step 6 of training algorithm.

Table 7.2 List of Modules

7.3.7 Implementation:

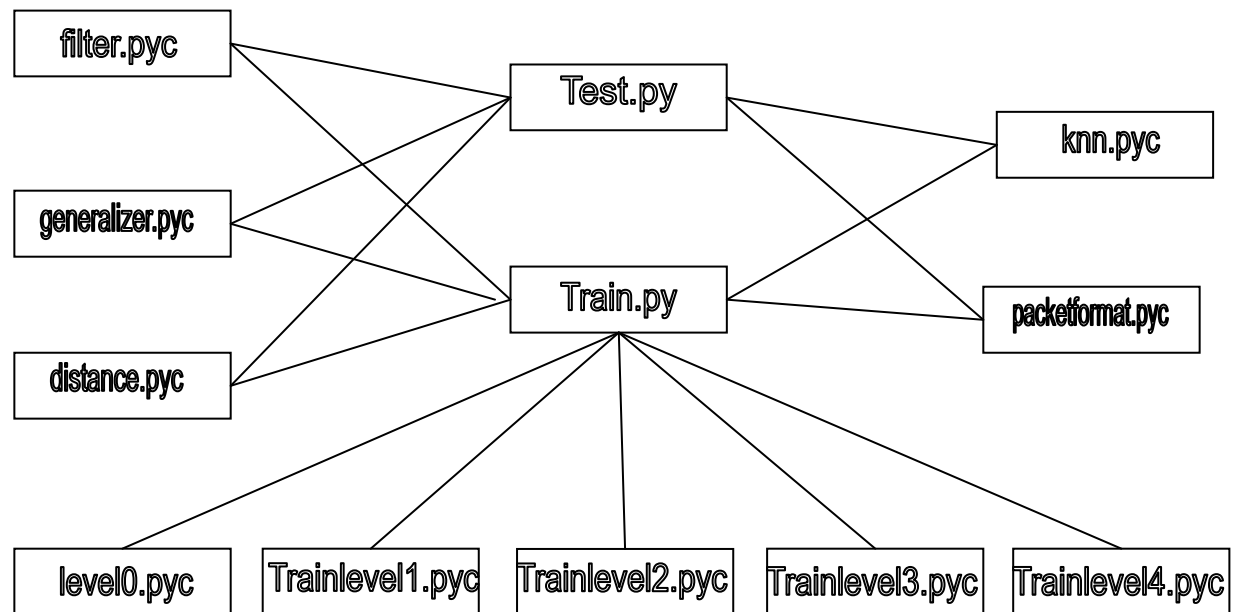


Fig. 7.2 Implementation Structure of Modules

7.3.8 Testing:

In testing phase, we determine the types of packets given in the testing file from one of the previously trained sessions.

Initially, user has to select one of the previously trained sessions, then system asks user to select testing file or enter testing file path. This file should be in correct format. After entering file path, system checks following conditions in sequence:

1. Check availability of file.
2. If file is available then check its format.
3. If format is correct then checks format of every data-entry.

If any error occurred then it prints proper error message and asks user to select file again.

Perform generalization and filter on testing data. Generalization changes the packet types from sub-attacks to attack and stores the data-entries back into testing data. Filter deletes duplicate entries from testing data and stores it back into testing data.

Before testing begins, testing data-entries are generalized and filtered and then pass this new testing data-set to testing algorithm.

7.3.9 Testing Algorithm:

Steps No.	Steps
1	Get parameters of training session into knndb_format, knn_classifier, and weighting function.
2	Print knndb_format, knn_classifier, and weighting function.
3	Get contents of knndb of training session into 'knndb'. Assign file pointer testingdb to read contents of given test data-set file. Initialize, correct_count["normal":0, "probe":0, "dos":0, "u2r":0, "r2l":0] and incorrect_count["normal":0, "probe":0, "dos":0, "u2r":0, "r2l":0]
4	For each 'entry' in testingdb: Classify entry using knndb
4.1	If correctly classified then Increment correct_count[entry_type]
4.2	Else Increment incorrect_count[entry_type]
	Save correct_count, incorrect_count into result.txt
5	Quit

7.3.10 Naïve Bayes

7.3.10.1 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.

7.3.10.2 Testing Algorithm

Steps No.	Steps
1	Read testing data packets line by line till end of file
1.1	Look up the table for prior value of every attribute value observed in testing
1.2	Calculate probability values for every outcome of attack recognized during training by (multiplying all prior probability values of 41 features * number of times number of times attack appearing)/(number of times number of times attack appearing * number of testing packets)
1.3	Maximum value will be declared as attack

7.3.11 AdaBoost Algorithm

7.3.11.1 Introduction

Boosting has been a very successful technique for solving the two-class classification problem. In going from two-class to multi-class classification, most algorithms have been restricted to reducing the multi-class classification problem to multiple two-class problems. Adaboost is a method of producing a very accurate prediction rule by combining inaccurate rules of thumb. The main objective of Adaboost (Adaptive Boosting) is to Improve the efficiency of classification.

7.3.11.2 Algorithm

Steps Steps

No.

- 1 Input: A set of training samples with labels $\{(x_1, y_1), \dots, (x_N, y_N)\}$, a componentLearn algorithm, the number of cycles T.
- 2 Initialize: The weights of training samples: $w_i^1 = 1/N$, for all $i = 1, \dots, N$.
- 3 Do for $t=1, \dots, T$
 - 3.1 Use the component learn algorithm to train a component classifier, h_t , on the weighted training samples,
 - 3.2 Calculate the training error of h_t :

$$\varepsilon_t = \sum_{i=1}^N w_i^t, y_i \neq h_t(x_i)$$

- 3.3 Set weight for the component classifier h_t :

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

- 3.4 Set weight for the component classifier h_t :

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

- 3.5 Update the weights of training samples :

$$w_i^{t+1} = \frac{w_i^t \exp\{-\alpha_t y_i h_t(x_i)\}}{C_t},$$

where $i=1, \dots, N$ and C_t is a normalization constant, and

$$\sum_{i=1}^N w_i^{t+1} = 1.$$

- 4 Output:

$$f(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right).$$

7.4 Business Logic and Architecture

7.4.1 Business Logic

The work which is being done and implemented needs to be documented for the one's 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 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 IEEE papers studying such as "Improving the Accuracy of Intrusion Detection Systems by using the Combination of Machine Learning Approaches", "Evaluating Machine Learning Algorithms for Detecting Network Intrusions" and "Comparisons of Machine Learning Algorithms Performance in Detecting Network Intrusion". We also made paper in the IEEE format which was selected in International Conference "ICNCS 2012, Hong Kong".

Our project is build and developed using three algorithms named K-NN, Naïve Bayes and AdaBoost. Working of K-NN 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 and AdaBoost can also be considered foe same matter. This was regarding only 3 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.

7.4.2 Architecture

Our project name is “To reduce False Alarm Rate (FAR) in Intrusion Detection System through Machine Learning Algorithm”. FAR is False Alarm Rate which means indicating a wrong signal for correct one and correct signal for the wrong one. Machine Learning means basically giving intelligence to the machine to learn itself just like how the kid starts learning itself 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. Intrusion is nothing but disturbance or an obstacle one can say. 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. 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 from 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.

3 Algorithms are being used for identifying attack major attacks DOS, Probe U2r, R2l and normal. KNN, Naïve Bayes and AdaBoost are three 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 8

Software testing

8.1 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.

8.2 Test Specification

8.2.1 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.

8.2.2 Test Plan

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 format	No
4	Input training data	Other than KDDcup dataset	Correct data	Wrong Dataset	No
5	Selection of Algorithm	Algorithm	Algorithm Selected	Algorithm Selected	Yes

6	Selection of Algorithm	None	Algorithm Selected	None Algorithm Selected	No
7	Input testing data	KDDcup dataset	Correct data	Correct data	Yes
8	Input testing data	KDDcup dataset	Correct data	Incorrect Feature Values	No
9	Input testing data	KDDcup dataset	Correct data	Dataset not in correct format	No
10	Input testing data	Other than KDDcup dataset	Correct data	Wrong Dataset	No

8.2.3 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.

8.2.4 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**

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 or from the external by system (ILF).

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**

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).

External Outputs

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

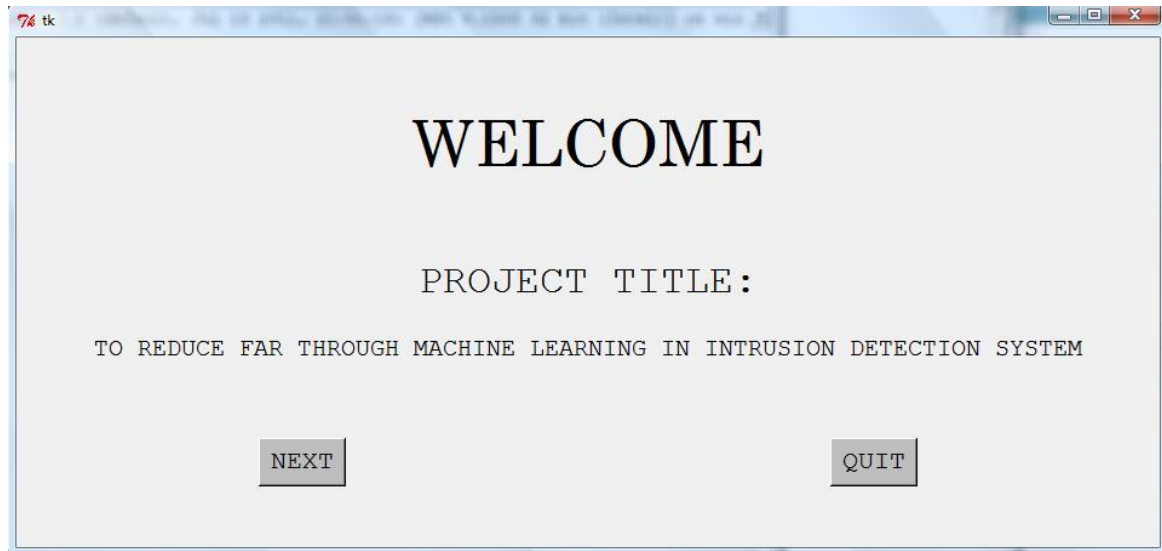
External Inquiries

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).

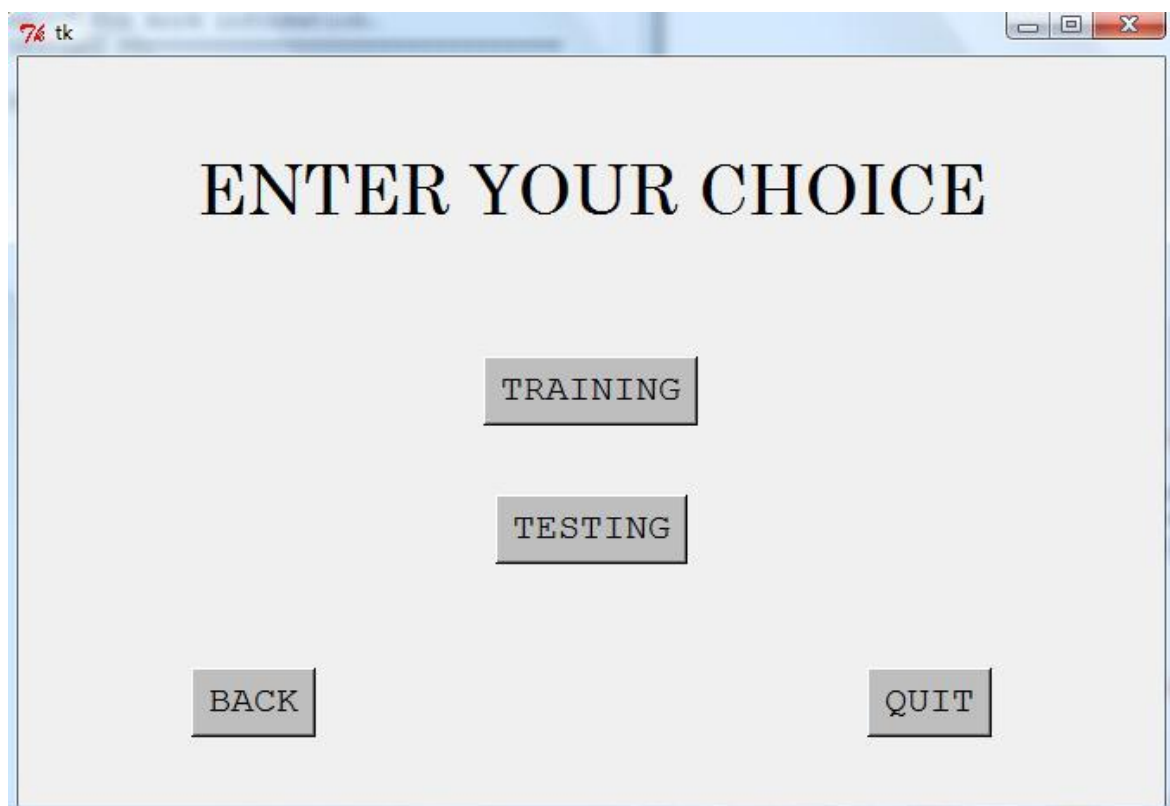
Chapter 9

Results

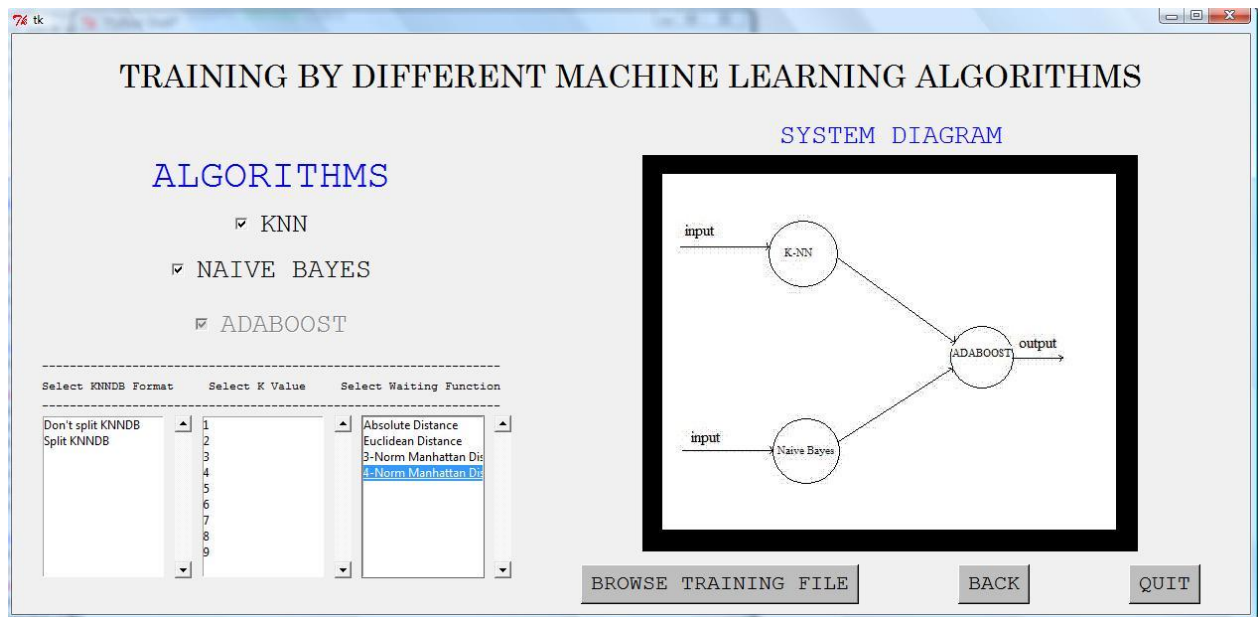
1. Welcome Screen:



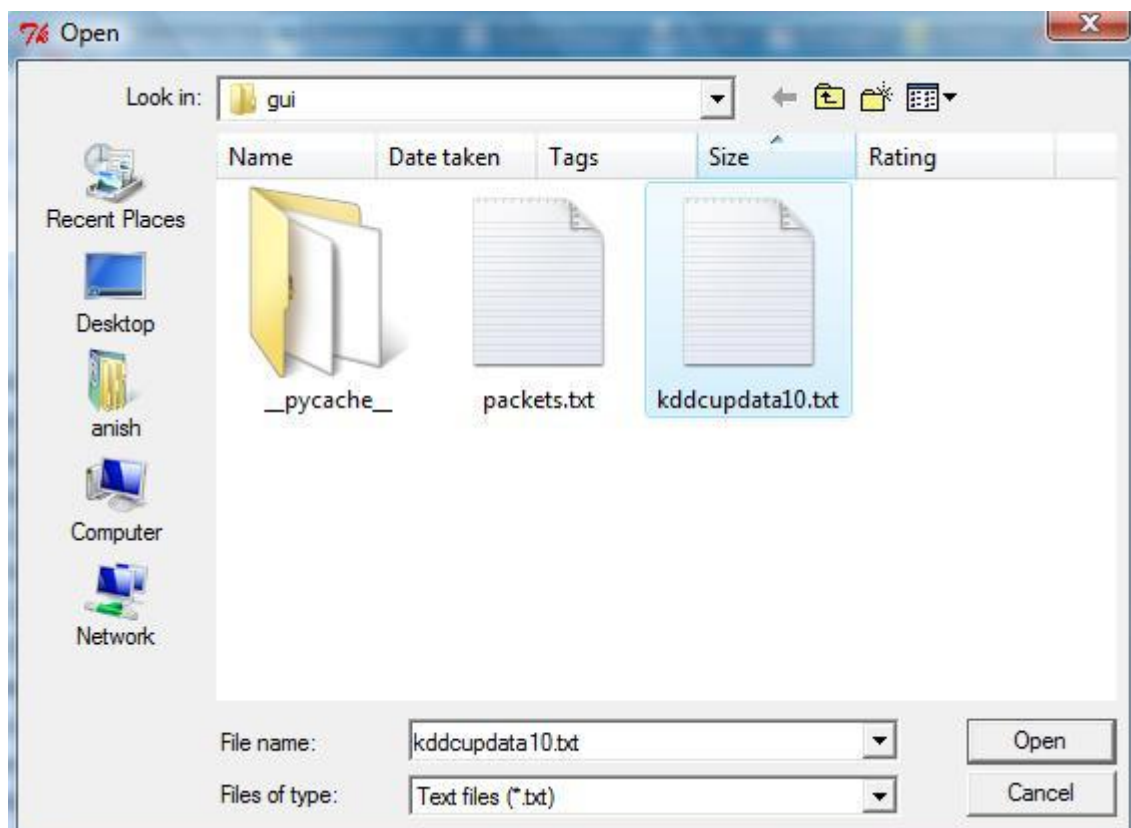
2. Training/Testing Selection:



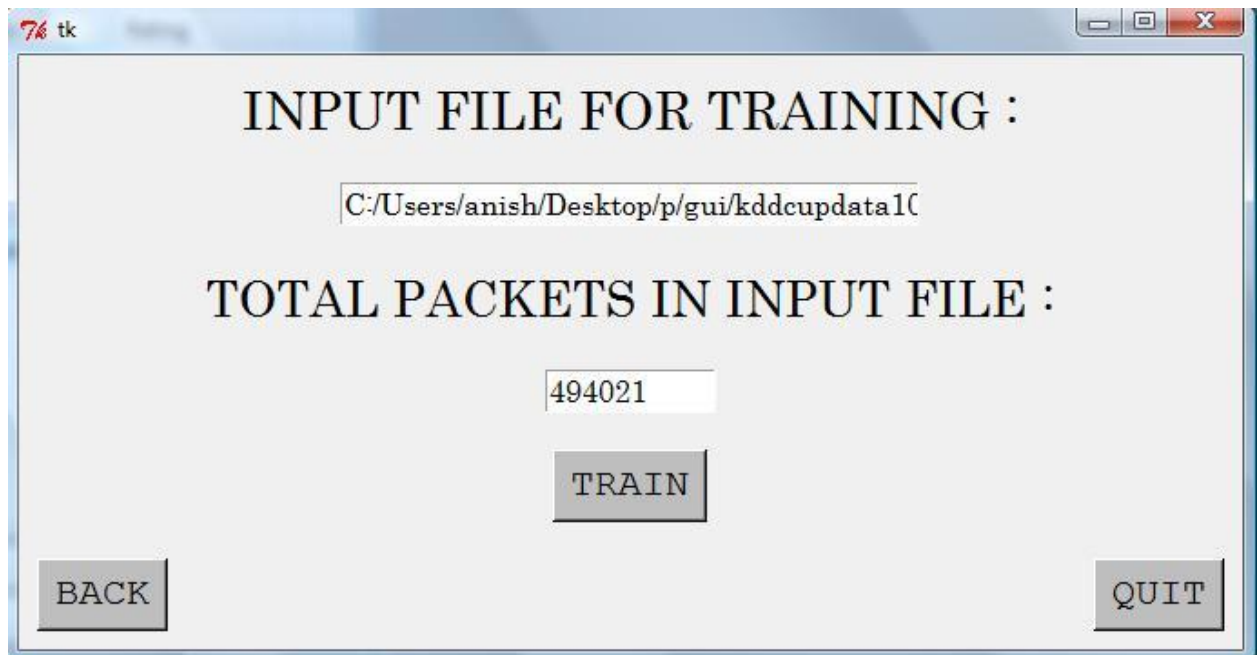
3. Training Algorithms Selection:



4. Browse KDD Data Set:



5. Input File Selection for Training:



tk

INPUT FILE FOR TRAINING :

C:/Users/anish/Desktop/p/gui/kddcupdata10

TOTAL PACKETS IN INPUT FILE :

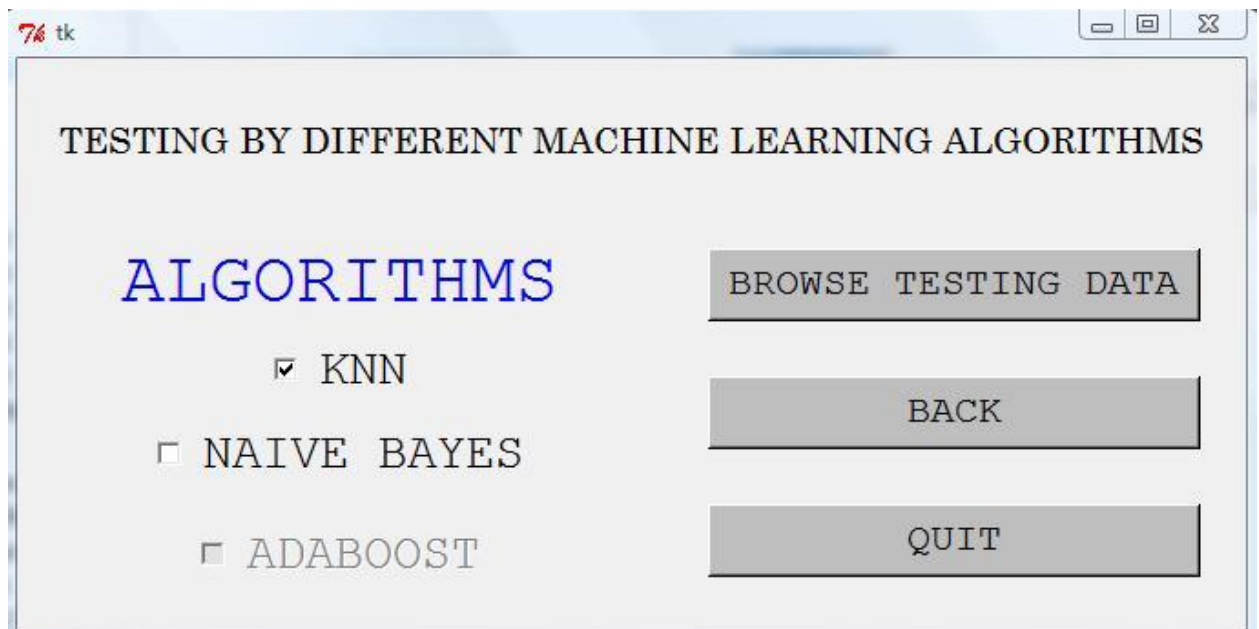
494021

TRAIN

BACK

QUIT

6. Browse Testing Algorithms and File Selection::



tk

TESTING BY DIFFERENT MACHINE LEARNING ALGORITHMS

ALGORITHMS

☒ KNN

☐ NAIVE BAYES

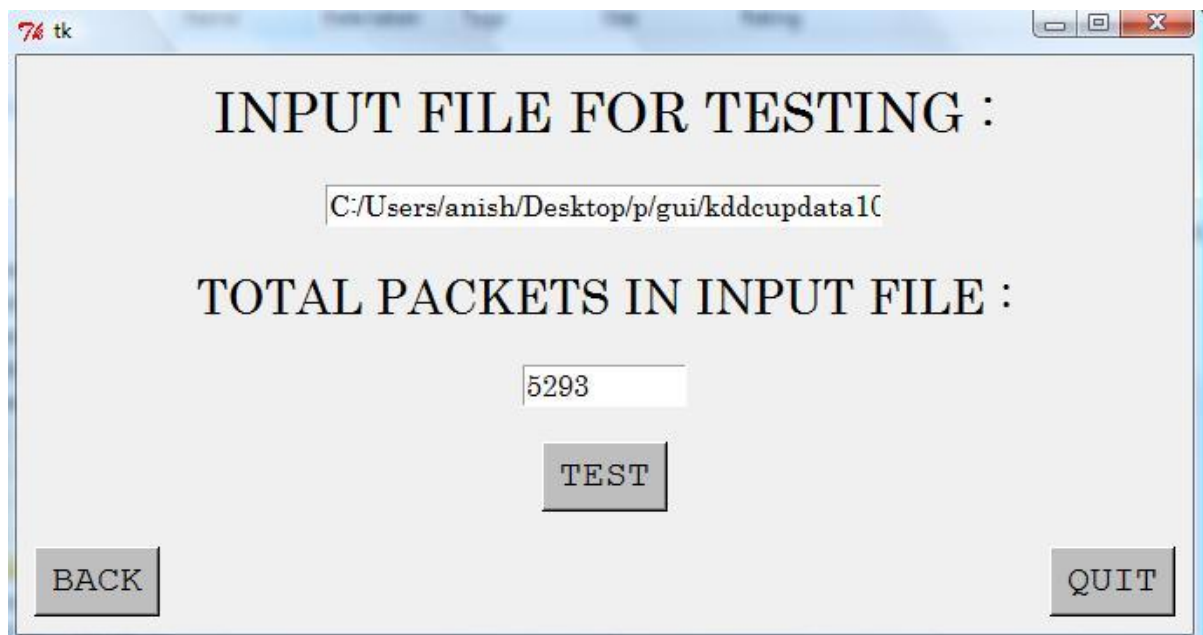
☐ ADABOOST

BROWSE TESTING DATA

BACK

QUIT

7. Input File Selection for Testing:



INPUT FILE FOR TESTING :

C:/Users/anish/Desktop/p/gui/kddcupdata10

TOTAL PACKETS IN INPUT FILE :

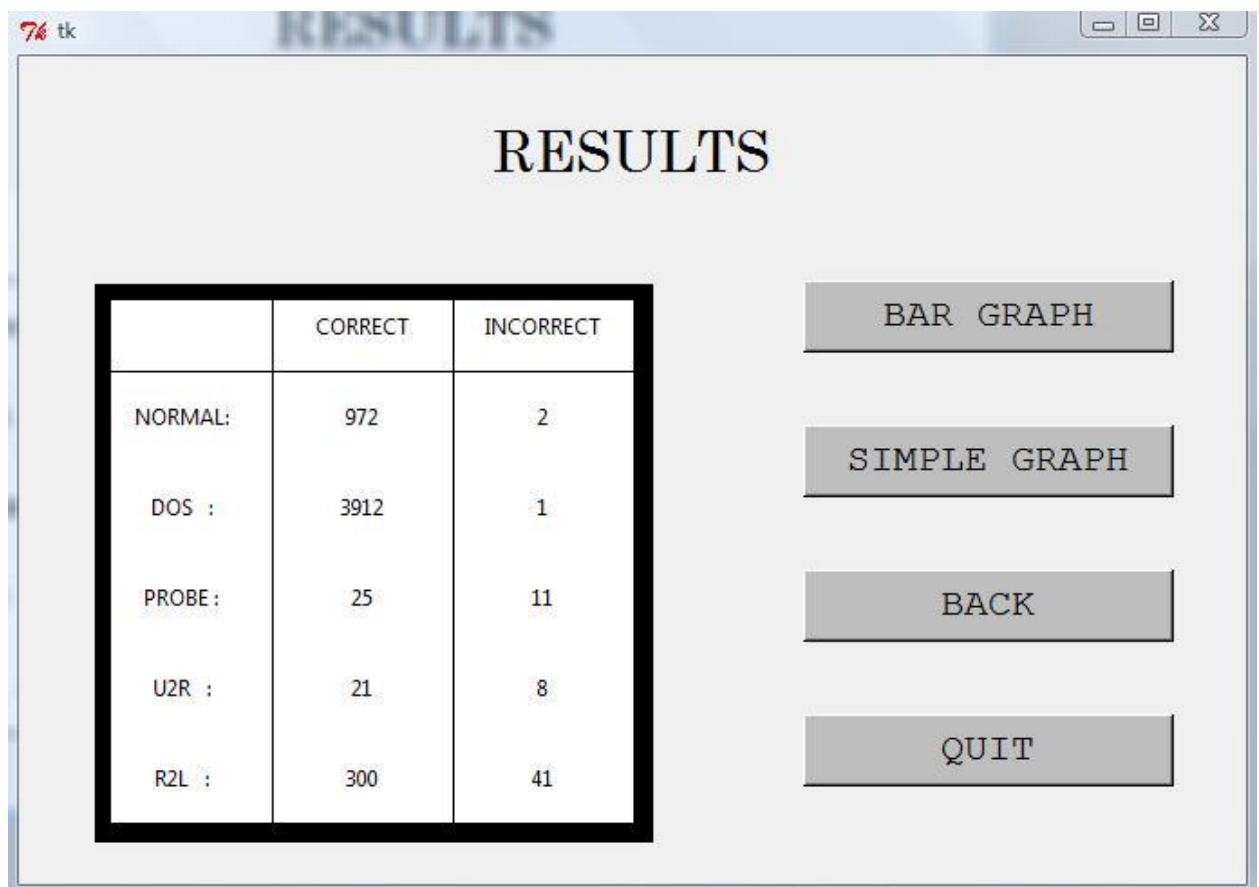
5293

TEST

BACK

QUIT

8. Result Display in Tabular Format:



RESULTS

	CORRECT	INCORRECT
NORMAL:	972	2
DOS :	3912	1
PROBE :	25	11
U2R :	21	8
R2L :	300	41

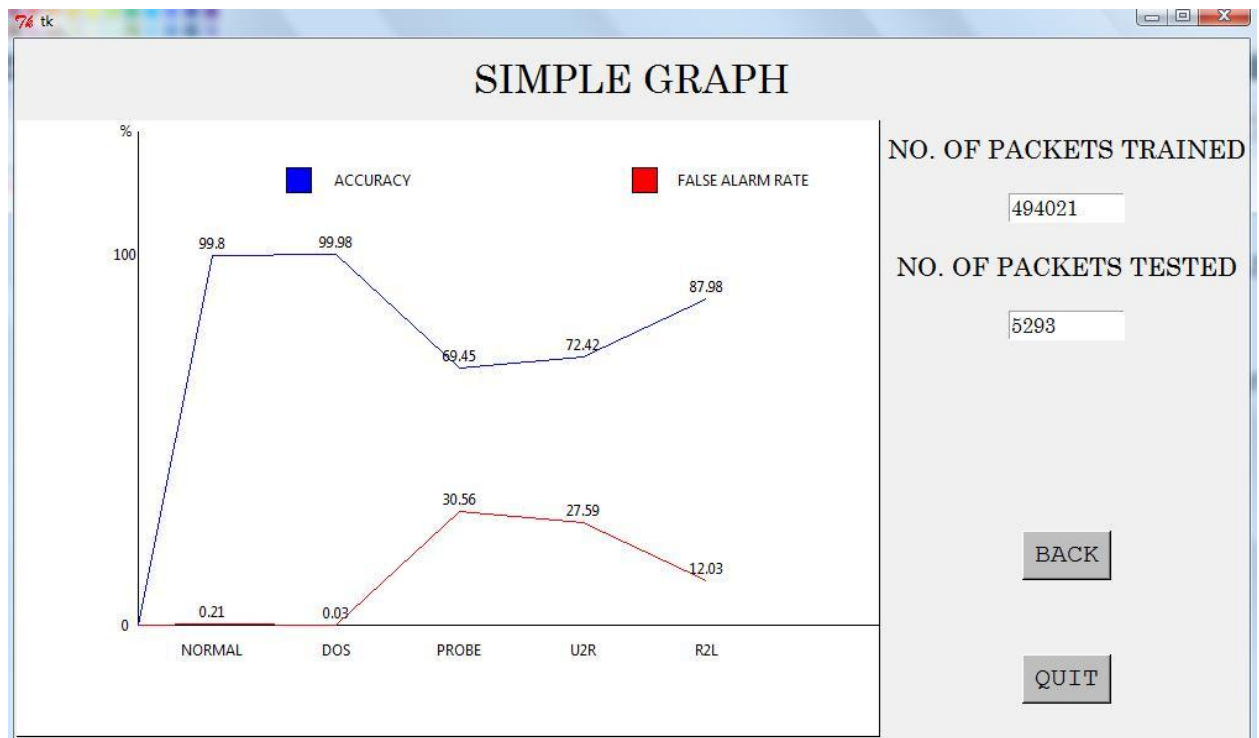
BAR GRAPH

SIMPLE GRAPH

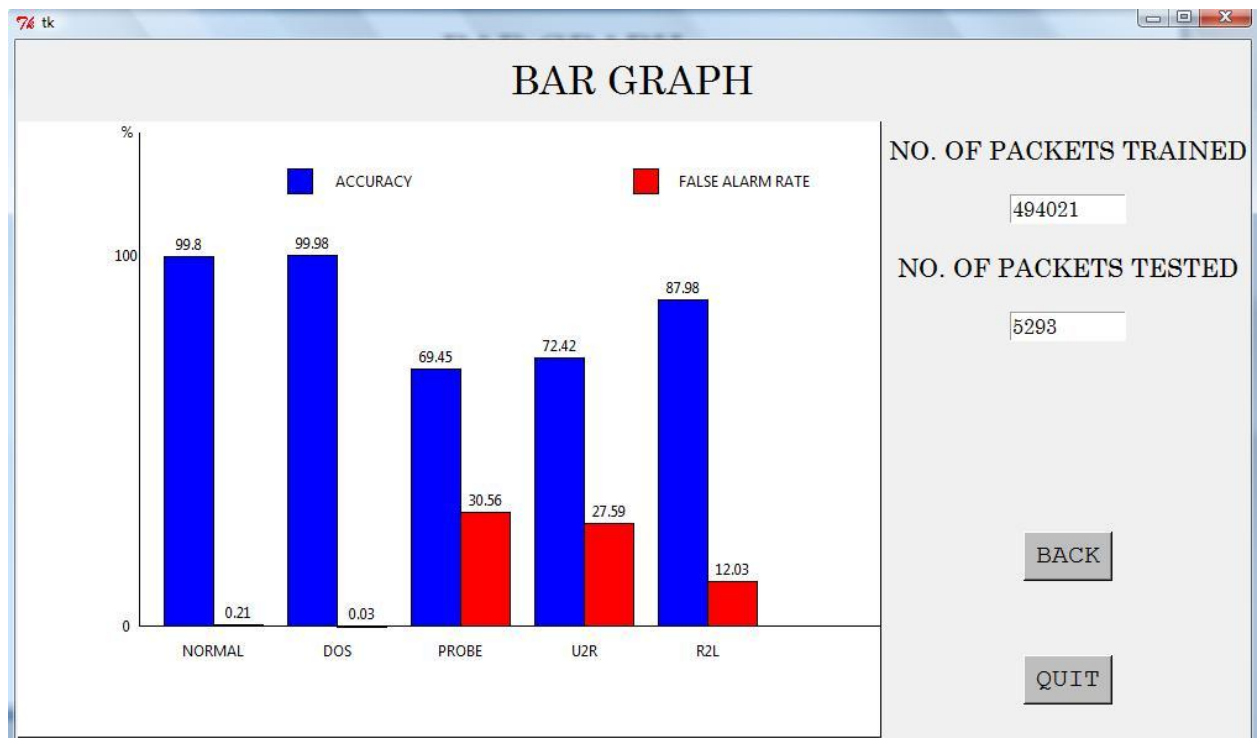
BACK

QUIT

9. Result Display in Graphical Format:



10.Result Display in Bar Graph Format:



Chapter 10

Deployment and Maintenance

10.1 User Help

10.1.1 Getting Started

Compile and Run the GUI main program. The welcome page will appear with python shell running at the back. The welcome shows the Project title with option 'NEXT' and 'EXIT'.

10.1.2 Training\Testing

This page contains two main button 'TRAINING' and 'TESTING' which will lead you to the training phase and testing phase respectively. There is 'BACK' and 'EXIT' button on each page.

10.1.3 Training Algorithm

After you click on 'TRAINING' button the page with different training algorithm appears. This page contains three option buttons namely 'k-NN', 'Naïve Bayes' and 'AdaBoost'. Option AdaBoost is disabled which gets selected if both k-NN and Naïve Bayes option is chosen. If k-NN option is selected then its three features listed in three list box also needs to be selected by double clicking on the required value. At right side of the page system diagram is displayed according to the option chosen. After selecting the options, training data needs to be browsed by clicking the 'BROWSE TRAINING DATA' button.

10.1.4 Browse Training File

Here the training data is selected (e.g. Kddcupdata10.txt) and open is clicked. If selected file is not in kddcupdata format or no file is selected, then error message popup and you need to again select the correct file.

10.1.5 Train

This page displays training file and number of packets it contains. The 'TRAIN' button activates the training algorithm and the training result is saved. After training the form with 'TRAINING' and 'TESTING' button is again displayed.

10.1.6 Testing Algorithm

After you click on 'TESTING' button the page with different training algorithm appears. This page contains three option buttons namely 'k-NN', 'Naïve Bayes' and 'AdaBoost'. Option AdaBoost is disabled which gets selected if both k-NN and Naïve Bayes option is chosen. After selecting the options, testing data needs to be browsed by clicking the 'BROWSE TESTING DATA' button which displays similar browsing page.

10.1.7 Test

This page displays testing file and number of packets it contains. The 'TEST' button activates the testing algorithm and the testing result is saved. After testing the Result form appears.

10.1.8 Result

This page displays result in tabular view. The results can also be obtained in graphical and bar graph format by clicking 'SIMPLE GRAPH' and 'BAR GRAPH' button respectively.

10.1.9 Exit

User can run different algorithms and compare their results.

Chapter 11

Conclusion and Future Scope

11.1 Conclusion:

In our project, we use different machine learning algorithms to model Intrusion Detection System (IDS) to improve detection rate and reduce False Alarm Rate (FAR). With the help of k-NN, AdaBoost, Naïve Bayes we have developed our algorithms and studied their performance. We have used KDD Cup99 as our data set which 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 (FAR) and accuracy for four categories of attack and normal data. Our aim is to reduce False Alarm Rate (FAR) and increase the detection rate compared to results of IEEE papers we had selected in our literature survey, in which we have succeeded.

11.2 Future Scope

Future works of our project are:

- 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.

Chapter 12

References

1. Kamarularifin Abd Jali, Muhammad Hilmi Kamarudin, Mohamad Noorman Masrek, "Comparision of Machine learning Algorithms Performance in Detecting Network Intrusion", 2011 International Conference on Networking and Information Technology.
2. Hadi Sarvari, Mohammad Mehdi Keikha, "Improving the accuracy of Intrusion Detection Systems by Using the Combinations of Machine Learning Approaches", 2011 IEEE.
3. Mrutyunjaya Panda, Manas Ranjan Patra, "Evaluating Machine Learning Algorithm for detecting Network Intrusions", International Journal of Recent trends in Engineering, Vol. 1, No. 1, May 2009.
4. "The UCI KDD Archive: KDD Cup 1999 Data Set,"<http://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data>.
5. J. McHugh, "Testing Intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratories", ACM Transactions on Information and System Security, vol. 3, no.4, pp. 262–294, November 2000.
6. Base, R. and Mell, P., "Intrusion Detection Systems", NIST Special Publications, sp800, 31-Nov-2001.
7. O. Depren, M. Topallar, E. Anarim, and M. KemalCiliz, An Intelligent intrusion detection system(IDS) for anomaly and misuse detection in computer networks. Expert systems with applications, Volume 29, Issue 4, pp. 713-722, November 2006.
8. A. Osareh, B.Shadgar, "Intrusion Detection in Computer Network based on Machine Learning Algorithm", IJCSNS International Journal of Computer Science and Network Security, Vol.8 No.11, November 2008.
M. V. Mahoney and P. K. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection", Proceedings of Recent Advances in Intrusion Detection, 2003.

Appendix A

- | | |
|---------|------------------------------------|
| 1. IDS | Intrusion Detection System. |
| 2. FAR | False Alarm Rate |
| 3. KDD | Knowledge Discovery Data |
| 4. k-NN | k Nearest Neighbour |
| 5. DOS | Denial of Service |
| 6. R2L | Remote to Local |
| 7. U2R | User to Root |
| 8. NIDS | Network Intrusion Detection System |